inf

Ministerul Educației și Învățămîntului
Casa Universitarilor Timișoara

inf nr. 2/88

buletin al CLUBULUI PROGRAMATORILOR

Colectivul de redacție
conf dr ing Crișan Strugaru
ș.l. ing. Ștefan Holban
ș.l. dr. ing. Ionel Jian
ing. Constantin Cozmiuc
Coperta, tehnoredactarea și prezentarea grafică
Eweline Thierjung
Cristian Bîrloncea

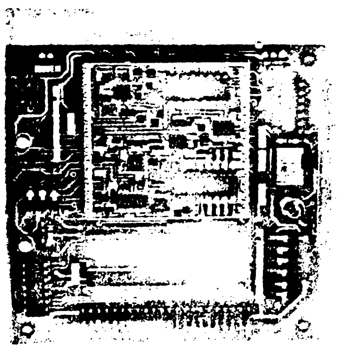
SUMAR

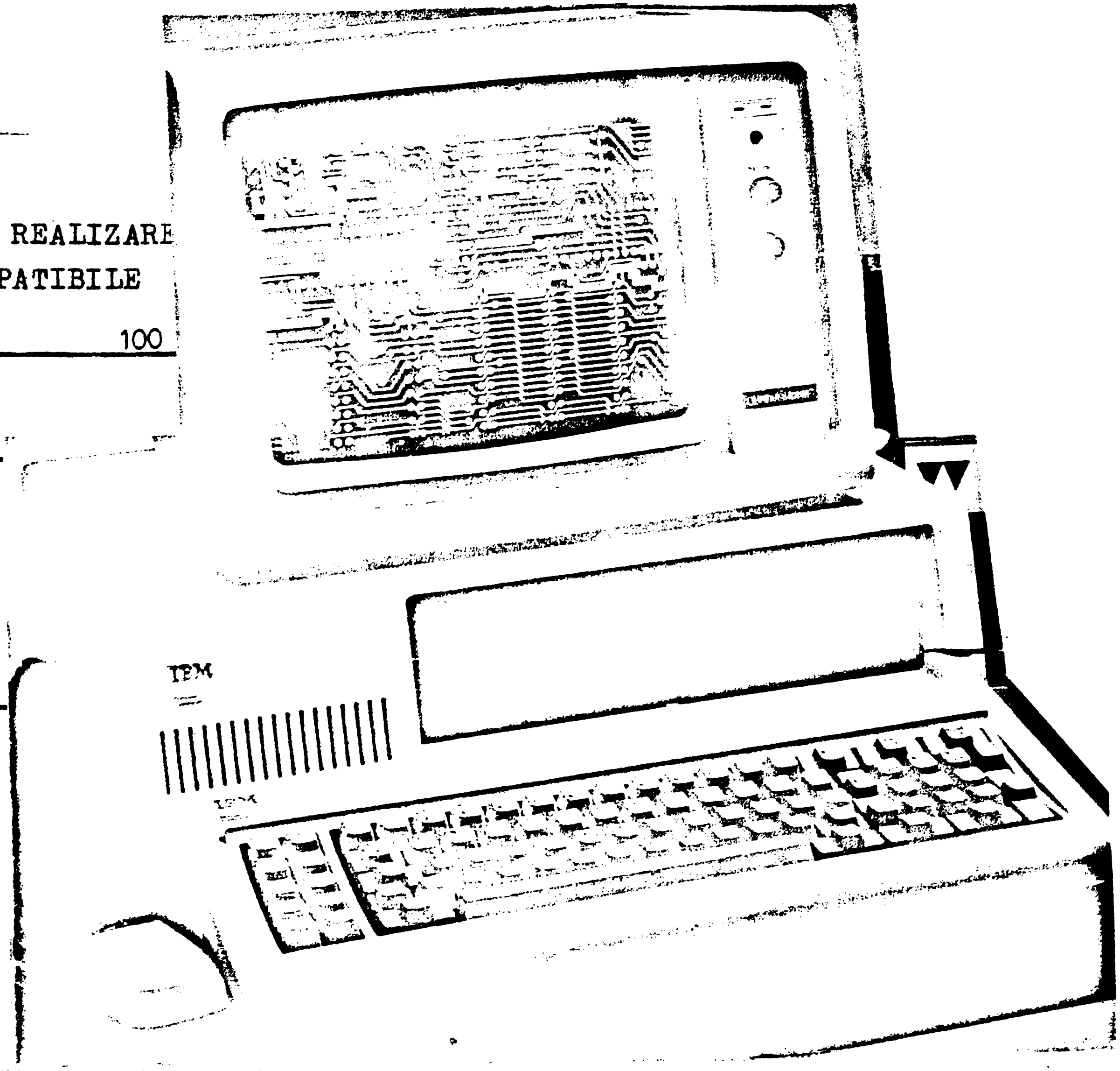
CALCULATORUL ÎN SPRIJINUL DUMNEAVOASTRĂ

MANUAL DE UTILIZARE

• PROGRAME •

• DIVERSE •





AS.ING. CEZAR MORUN
ING. DANIELA BRAD
ING. HARALD SCHRIMPF

SISTEMUL DE OPERARE TIM-S V2

1.Introducere

La mai bine de 1 an de la lansarea microcalculatorului TIM-S, purtind discutii cu diferiti utilizatori ai acestuia, a inceput sa se contureze necesitatea efectuarii unor modificari in sistemul de operare.

1.1. Sistemul de operare TIM-S V1 sustine iesire grafica pe imprimantele SCAMP CDC 9335 si MIM-40 dar majoritatea TIM-S-urilor au fost livrate cu imprimante SCAMP CDC 9335, ROBOTRON 6311 (ROMOM), ROBOTRON 6313. Ulterior s-a incercat adaptarea SO TIM-S V1 la aceste imprimante rezultind o varietate de SO pe punctul de a iesi de sub controlul realizatorilor. SO TIM-S V2 realizeaza o tratare unitara a imprimantelor enumerate mai sus prin LPRINT grafic, LPRINT ASCII si COPY.

1.2. Multi utilizatori si-au exprmat dorinta unui rind mai lat atit pe ecran cit si la imprimanta.

Ecranul, datorita structurii hardware, a putut fi extins la maxim 64 caractere /rind. Prin aceasta realizare in SO TIM-S V2 s-a pregatit in acelasi timp implementarea pe microcalculatorul TIM-S a SO CP/M.

La imprimanta poate fi selectata o latime a rindului de pina la 255 caractere(teoretic) in regim LPRINT ASCII, LPRINT grafic raminind la latimea de 32 caractere/rind.

Toate iesirile pe ecran se fac prin canalul "S", iar la imprimanta prin canalul "P", de fapt o crestere a compatibilitatii SO TIM-S V2 fata de SO ZX Spectrum (s-a renuntat la canalele "A","a","B","b","X","x","G","g","M","m").

Selectarea optiunilor pentru ecran si imprimanta se realizeaza cu ajutorul instructiunii FORMAT (neutilizata in SO ZX Spectrum 48K).

1.3. Primii 16 KO RAM nu au fost practic folositi decit pentru memorarea SO din ROM pentru a putea lucra la o frecventa de tact de 6 MHz. Totusi aceasta posibilitate a fost rar folosita datorita lipsei de microprocesoare Z80B.



Sistemul de operare TIM-S V2 foloseste in mod rational aceasta zona de memorie cu ajutorul instructiunii POKE si a functiilor PEEK si USR care vad toti cei 80 kO memorie (16 kO ROM + 64 kO RAM) cit si a instructiunii MOVE (neutilizata nici ea in cadrul SO ZX Spectrum 48K).

1.4. S-au inlaturat doua erori din rutinele de afisare si s-a corectat functia SCREEN$ care functioneaza acum corect.

PRINT CHR$(9) desi nu semnaliza eroare, nu avea nici un efect in afisare; acum muta cursorul cu o coloana la dreapta.

PRINT CHR$(8) functiona corect doar pina in coloana 0, rindul 1; acum se poate executa BS (Back Space) pina in coloana 0, rindul 0.

2.Diferente intre SO TIM-S V2 si SO ZX Spectrum 48K

2.1 Instructiunea FORMAT:

FORMAT "nume",NC:nr.1,BD:nr.2,G:nr.3

nume: "S","s" -ecran
"D","d" -imprimanta SCAMP CDC 9335
"E","e" -imprimanta ROBOTRON 6311
"F","f" -imprimanta ROBOTRON 6313
"G","g" -imprimanta utilizator

Utilizatorul isi va scrie propria rutina de dialog cu imprimanta si va plasa in locatiile 23728(#5CB0) -octetul low si 23729(#5CB1) -octetul high adresa acestei rutine .Aceasta metoda se poate folosi pentru tiparirea in regim ASCII.

nr.1: - numar reprezentabil pe un octet (cuprins in intervalul 0-255).
- da numarul de caractere pe rind de ecran sau imprimanta.

Obs.2.1.1 -pentru nume =S sau s pentru 0=<nr.1=<32 se va afisa cu 32 caractere /rind iar pentru nr.1=>33 se va afisa cu 64 caractere/rind.

Obs.2.1.2 -pentru imprimanta dupa tiparirea a nr.1 caractere pe rind SO insereaza automat caracterele CR (#0D) si LF (#0A).

nr.2: - trebuie sa fie un numar din urmatoarea lista: 0, 50, 110, 300, 600, 1200, 2400, 4800, 9600, 19200.
- 0 va selecta dialog cu imprimanta prin intermediul interfetei paralele tip CENTRONICS.
- celelalte valori vor selecta BauD rate pentru dialogul cu imprimanta prin intermediul interfetei seriale tip RS232C.

nr.3: - 0 selecteaza regim ASCII pentru imprimante.
- 1 selecteaza regim grafic pentru imprimante.
- de fapt nr.3 poate fi un numar reprezentabil pe un octet (0-255) - un numar par selectind regim ASCII iar un numar impar selectind regim grafic.

Obs.2.1.3. Dupa initializare este selectat regimul de afisare cu 32 caractere/rind de ecran si imprimanta, dialog prin intermediul interfetei paralel cu imprimanta SCAMP CDC 9335 in regim grafic.

2.1.4. Selectarea parametrilor pentru ecran nu influenteaza parametrii pentru imprimanta si viceversa.

2.1.5 Pentru ecran parametrii BD si G sint nesemnificativi, dar obligatorii (din pacate).

2.1.6 Selectarea tipului imprimantei prin "nume" este necesara doar pentru regimul grafic, pentru a trimite caracterele de control specifice.

2.2. Instructiunea PRINT

2.2.1. Regimul de afisare cu 32 caractere/rind.

In acest regim nu exista nici o deosebire fata de ZX Spectrum cu exceptia corectarii celor 2 erori amintite in capitolul introductiv.

2.2.2. Regimul de afisare cu 64 caractere/rind.

In acest regim de afisare exista urmatoarele particularitati:

PRINT AT R,C
- C poate lua valori in domeniul 0-63

PRINT TAB C
- C va fi redus la Cmod64

PRINT cu ,
- va afisa in coloanele 0, 16, 32, 48

Obs.2.2.2.1 Rezolutia de atribute (INK, PAPER, BRIGHT, FLASH) ramine cea din regimul cu 32 caractere/rind.

2.2.2.2 Atributul pe o zona caracter 8*8 pixeli este "fixat" de ultimul caracter inscris in aceasta zona, fie in cele 4 coloane din stinga, fie in cele 4 coloane din dreapta zonei.

2.2.2.3 Desi editarea liniilor BASIC se poate face si in regim de afisare cu 64 caracter/rind pot aparea dificultati in urmarirea cursorului datorita celor mentionate in observatiile 2.2.2.1 si 2.2.2.2

2.2.2.4 INVERSE si OVER functioneaza pe matricea de 4*8 pixeli in care sint afisate caracterele.

2.2.2.5 Pe un acelasi rind se pot alterna ori de cite ori se doreste cele 2 regimuri de afisare (cu ajutorul instructiunii FORMAT). La trecerea din regimul 32 in regimul 64 numarul coloanei in care se va afisa urmatorul caracter se calculeaza cu formula:

C64 = C32 * 2

La trecerea din regimul 64 in regimul 32 numarul coloanei in care se va afisa urmatorul caracter se calculeaza cu formula:

C32 = INT ((C64 + 1) /2)

2.3. Instructiunea LPRINT

2.3.1. LPRINT grafic

In regim grafic LPRINT functioneaza ca si la ZX Spectrum cu imprimanta ZX Printer cu deosebirea ca buffer-ul de imprimanta se transfera prin doua treceri ale capului de tiparire prin fata hirtiei: la prima trecere se tiparesc cele 6 pixel-lines superiori ai zonei caracter iar la a doua trecere cele 2 pixel-lines inferiori plus inca 4 linii albe.

2.3.2. LPRINT ASCII

In acest regim lungimea unui rind la imprimanta poate fi selectata, functie de posibilitatile acesteia, pina la max. 255 caractere.

Se transmit spre imprimanta toate codurile ASCII #00 - #7F, o tratare speciala avind doar urmatoarele:

- #0D (CR) - automat dupa acest cod se mai emite si codul #0A (LF)
- #7F (DEL) - in locul acestui cod se transmit codurile caracterelor urmatorului sir: "(c)"

Codurile semigrafice si UDG se inlocuiesc cu codul #20 (SP).

Obs.2.3.2.1 Listarea programelor in acest regim poate produce surprize neasteptate datorita caracterelor de control Spectrum pentru afisare pe ecran care trec si ele spre imprimanta.

2.3. Instructiunea COPY

COPY [nr.]

nr. - numar reprezenatabil pe un octet (0-255)
- optional (la fel ca si la RUN sau LIST)

COPY (sau echivalentul COPY 0) va copia 176 pixel-lines (in regim grafic) sau 22 de rinduri (in regim ASCII) de pe ecran la imprimanta.

COPY 1 (de fapt orice numar #0) va copia 192 pixel-lines (in regim grafic) sau 24 de rinduri (in regim ASCII) de pe ecran la imprimanata.

Prima forma este identica cu COPY la ZX Spectrum si utila in unele programe cum ar fi

MASTERFILE sau SUPERCODE.

Cea de-a doua varianta copiaza intregul ecran, identic TIM-S V1, si este si ea utila in diferite programe de grafica care permit utilizarea intregii zone ecran (fereastra).

2.4. Instructiunea POKE

POKE nr.1,nr.2

Noutatea consta in faptul ca nr.1 este acum un numar cuprins in intervalul 0-131071 (128K).

Pentru 0=<nr.1=<65535 se lucreaza ca si in vechiul SO.

Pentru 65536=<nr.1=<131071 instructiunea POKE va adresa cei 64 kO RAM, indiferent din care pagina lucreaza SO. In acest fel se poate modifica continutul unor locatii din primii 16 kO RAM.

2.5. Functia PEEK

PEEK nr.

Domeniul de memorie adresat de nr. este acelasi cu cel adresat de nr.1 din instructiunea POKE.

2.6. Functia USR

USR nr.

Si pentru nr. sint valabile observatiile referitoare la nr.1 de la instructiunea POKE.

2.7. Instructiunea MOVE

MOVE nr.1,nr.2,nr.3

Aceasta instructiune transfera continutul unei zone de memorie in alta zona, avind la baza urmatorii parametrii:

nr.1: - adresa memorie zona sursa
nr.2: - adresa memorie zona destinatie
nr.3: - lungimea zonei care se transfera

Obs.2.7.1 Parametrii nr.1 si nr.2 se refera in exclusivitate la cei 64 kO RAM.

2.7.2 nr.2 + nr.3 nu trebuie sa depaseasca granita celor 64 kO.

2.7.3 Zona de memorie RAMO dintre adresele #3C00 (15360) - #3CFF (15615) este protejata la scriere atit pentru instructiunea MOVE cit si instructiunea POKE, aceasta zona continind 4 variabile de sistem noi cit si rutine a caror existenta in RAM este indispensabila pentru buna functionare a sistemului.

2.8. TIM-S EXT1

Daca se lucreaza cu SO din ROM in primii 10 kO RAM se poate incarca de pe caseta softul TIM-S EXT1. In acest mod pe un microcalculator TIM-S fara extensia TIM-S EXT1 se pot deja elabora programe care presupun existenta acestei extensii.

Bibliografie:

1. Manual de utilizare TIM-S
2. Manual de utilizare TIM-S EXT1
3. The Complete Spectrum ROM Disassembly - Ian Logan, Frank O'Hara
4. The Spectrum Operating System - Steve Kramer
5. Proiect de diploma 1988 - Daniela Brad, Harald Schrimpf

S.L.ING. MESAROS-ANGHEL VOICU
ING. PUTERITY MIODRAG

EXTINDEREA INTERPRETORULUI BASIC LA COMPUTERELE COMPATIBILE SPECTRUM

- exemplu de extindere a posibilitatilor grafice -
prin "dublarea rezolutiei"

0. GENERALITATI

Problema extinderii interpretorului BASIC se pune in special atunci cind este necesar ca anumite subrutine in cod masina sa fie folosite frecvent iar forma "clasica":

[{POKE <adresa>,<parametru>:}] RANDOMIZE USR <adresa_start>

ar dauna vitezei de introducere a textului BASIC precum si claritatii/elegantei acestuia.

In vasta biblioteca de software a Sinclair SPECTRUM-ului exista deja citeva programe utilitare care realizeaza extinderea interpretorului BASIC (BLAST COMPILER, COLT COMPILER, BASIC EXTENDU, BETA BASIC s.a.m.d.).

Prezentul articol isi propune sa descrie o noua metoda de extindere a interpretorului BASIC si sa o aplice unui caz concret, oferind o metoda de sporire a posibilitatilor grafice la calculatoarele compatibile SPECTRUM, prin DUBLAREA REZOLUTIEI la copierea pe imprimanta a memoriei video.

1. INTERPRETORUL BASIC SI METODE DE EXTINDERE

1.1. INTERPRETORUL BASIC

La calculatoarele compatibile SPECTRUM, modul de functionare al interpretorului BASIC este urmatorul:

- utilizatorul introduce de la tastatura o "tentativa" de linie BASIC in zona de editare (care incepe la adresa data de variabila de sistem E LINE (23641)). Aceasta este afisata in

partea de jos a ecranului cu cuvintele cheie tiparite in forma lor literala.

- tastind ENTER (ASCII 13), se semnaleaza sfirsitul liniei si se trece la verificarea ei sintactica.

- DACA sintaxa liniei este corecta, ATUNCI ea este copiata din zona de editare in zona de program BASIC (care incepe la adresa data de variabila de sistem PROG (23635)) corespunzator numarului de linie SAU daca acesta lipseste, se executa linia ca si o comanda imediata. De asemenea in aceasta faza are loc si evaluarea constantelor numerice care in zona de program apar ca insiruirea de coduri ASCII corespunzatoare cifrelor constantei urmata de codul de control (ASCII 14) si de cinci octeti care formeaza reprezentarea in virgula flotanta a constantei.

- ALTFEL linia este afisata din nou in partea de jos a ecranului (consola) dar cu markerul de eroare ("?" pilpiitor) indicind locul unde a fost detectata eroarea. Adresa caracterului de dupa markerul de eroare este data de variabila de sistem XPTR (23647).

- odata ce a fost introdus, un program BASIC poate fi lansat in executie, fapt semnalizat de starea bitului 7 din variabila de sistem FLAGS (23611). Acesta este resetat pentru verificarea sintactica si setat pentru linia in executie.

Din punct de vedere soft, cele doua etape sint partial suprapuse. Mecanismul este oarecum redundant deoarece verificarea sintactica se face si la rularea programului. In acest caz:

- DACA sintaxa unei instructiuni este corecta, ATUNCI se evalueaza si valoarea (nu doar tipul) expresiilor care dau parametrii instructiunii (daca exista) si se executa subrutina de comanda a instructiunii. Apoi se trece la interpretarea urmatoarei instructiuni.

- ALTFEL interpretorul detecteaza o eroare de tip NONSENSE IN BASIC (codul erorii se gaseste in variabila de sistem ERRNR (23610)) iar programul se opreste, revenindu-se in modul de editare. Modul de editare este semnalat de bitul 5 din variabila de sistem FLAGX (23665). Acesta este resetat pentru modul de editare si setat pentru modul INPUT.

Atit la verificarea sintactica cit si in executie, detectarea unei erori duce la o secventa de tipul:

```
RST  #08 ; subrutina de tratare a erorilor BASIC
DEFB <cod_eroare>
```

Aceasta subrutina face ca stiva masinii sa fie incarcata cu valoarea continuta in variabila de sistem ERRSP (23613) iar variabila de sistem ERRNR (23610) cu codul erorii. In acest moment, pe stiva se afla o adresa care este intotdeauna #1303 cind subrutina de tratare a erorii a fost chemata din timpul executiei. Aceasta subrutina poate fi apelata din editor sau din analizorul sintactic, caz in care pe stiva se vor gasi alte valori. Subrutina de tratare a erorii se termina cu un RET care dupa cum se stie face ca registrul PC sa fie incarcat cu adresa luata de pe stiva (deci programul continua de la adresa respectiva).

Pentru a permite intelegera capitolelor care urmeaza este necesar sa mai amintim ca:

- in general, linia este parcursa cu ajutorul subrutinelor GET_CH (RST #18) si NXT_CH (RST #20) iar adresa caracterului curent este inmagazinata in variabila de sistem CHRADD (23645).

- numarul liniei aflate in executie este retinut de variabila de sistem PPC (23621) iar al instructiunii curente din cadrul liniei in variabila de sistem SUBPPC (23623);

1.2. EXTINDEREA INTERPRETORULUI BASIC

Noile instructiuni vor fi privite de interpretorul BASIC ca erori de tipul NONSENSE IN BASIC. Aceste erori trebuie interceptate si cercetate pentru a putea determina daca sint erori propriuzise sau sint extensii de BASIC. In acest ultim caz extensiile de BASIC trebuie verificate sintactic (conform

sintaxei impuse de utilizator), iar in executie trebuie chemate subrutinele de comanda a extensiilor (bineinteles, tot scrise de utilizator). In final, controlul trebuie redat interpretorului BASIC pentru a prelua instructiunea urmatoare.

Din studiul unor programe utilitare care realizeaza extinderea interpretorului BASIC s-au determinat urmatoarele metode:

1.2.1. BASIC EXTENDU de ERE INFORMATIQUE

propune o metoda foarte eleganta de interceptare a erorilor de tip NONSENSE IN BASIC.

Modificind corespunzator valoarea din ERRSP se pot intercepta orice erori. Erorile "veritabile" sint tratate in modul obisnuit. Daca sint indeplinite anumite conditii (ex. utilizatorul este in faza de editare, lungimea zonei de editare este nenula, ultima tasta apasata a fost ENTER, etc.) si functie de starea bitului 7 din FLAGS, se poate face o discutie care sa duca la tratarea corespunzatoare a noilor instructiuni.

O problema destul de dificila, rezolvata cu succes de programatorii de la ERE INFORMATIQUE consta in "aranjarea" corecta a pozitiei si structurii stivei, astfel incit erorile sa fie "deviate" intotdeauna la subrutina de selectare si discutie. Problema este dificila din cauza ca mecanismul de tratare al erorilor lucreaza cu stiva necompensata (nu se pun pe stiva un numar egal de octeti cu cei preluati) iar ERRSP puncteaza de regula in virful memoriei (uzual RAMTOP-2).

Sintaxa noilor instructiuni este de tipul:

<numar_linie> ! <instructiune> [{<parametru>,}]

Verificarea sintactica se face atit dupa editare cit si la rulare. Sint permise mai multe instructiuni pe aceeasi linie separate de ":". Daca sintaxa este corecta din punctul de vedere a extensiilor, se renunta la caracterul "!" (nu mai apare in zona de program). Noile instructiuni se tasteaza litera cu litera.

Dezavantajul principal al metodei rezida in discutia extrem de complicata care permite decelarea erorilor de extensii. Aceasta duce la o scadere de viteza in executia programului in general, nu numai in cazul extensiilor.

1.2.2. COLT COMPILER de HISOFT si BETA BASIC de BETASOFT

se folosesc tot de ERRSP pentru devierea tratarii erorii la iesirea din subrutina de la RST #08, in schimb eroarea din perioada de executie este detectata mult mai simplu. La adresa #1303 (pe care puncteaza in executie valoarea din ERRSP) se afla instructiunea HALT. Aceasta are ca efect oprirea microprocesorului pina la aparitia unei cereri de intrerupere. In mod normal, SPECTRUM-ul trateaza intreruperea in modul IM 1 care duce la un restart la adresa #38 unde se realizeaza scan-area tastaturii si incrementarea valorii din FRAMES (23768) pentru ceasul de timp real. Cererea de intrerupere este generata de ULA la fiecare 20 ms sincron cu generarea impulsului de sincronizare cadre din semnalul video. Z80-ul permite si un mod mult mai puternic de intrerupere, IM 2, in care adresa subrutinei de tratare a intreruperii se gaseste la adresa determinata prin concatenarea registrului I (partea semnificativa) cu magistrala de date. Daca vectorizarea tratarii intreruperii se face in RAM, avem un instrument foarte puternic de interceptare a erorii. Decelarea erorilor de extensii se face mult mai simplu si mai rapid decit in cazul precedent.

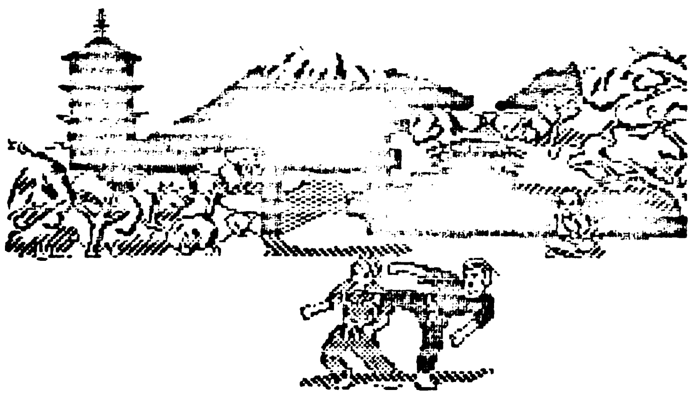
Deoarece SPECTRUM-ului ii pot fi atasate la EXPANSION PORT dispozitive diverse si deoarece unele din acestea nu decodifica corect semnalele de comanda (in favoarea simplitatii), continutul magistralei de date in momentul cererii de intrerupere nu poate fi stabilit (de regula el este #ff). In acest caz se foloseste o tabela de adrese care face ca subrutina de tratare a intreruperii sa fie independenta de partea nesemnificativa a adresei vectorului.

Extinderea interpretorului de BETA BASIC se face pe baza acestei metode insa este mult mai completa.

In cazul compilatorului COLT, documentatia acestuia descrie o modalitate de introducere a unor noi instructiuni de catre utilizator.

1.2.3. BLAST COMPILER de O.C.S.S.

isi propune sa extinda interpretorul BASIC cu citeva instructiuni sub forma unor REM-uri speciale (REM%,REM!,REM&). Unele dintre extensii sint directive ale compilatorului, altele etensii de BASIC. Metoda este mai simpla dar mai putin eleganta si se bazeaza pe modificarea subrutinei de comanda a instrucriunii REM.



1.2.4. INTERFATA ZX 1

permite o foarte simpla extindere a interpretorului BASIC bazata pe o trasatura a SHADOW ROM-ului (cei 8K ROM ai interfetei) care in scopul executarii noilor instructiuni pentru MICRODRIVE, interfata RS232 si LOCAL AREA NETWORK, intercepteaza HARD (!) erorile care conduc la un fetch de la adresa #0008. Dupa decelarea erorilor propriuzise de instructiunile noi specifice interfetei, are loc revenirea in interpretorul BASIC in ROM-ul de baza prin intermediul noii variabile de sistem VECTOR (23735) care in mod normal contine o adresa din ROM (#01F0). Alterind aceasta valoare se poate forta trecerea la subrutinele utilizatorului care realizeaza binecunoscuta separare a erorilor de extensii. Utilizatorul trebuie sa prevada un modul de verificare sintactica si un modul care contine subrutinele de comanda a noilor instructiuni.

2. O NOUA METODA DE EXTINDERE A INTERPRETORULUI BASIC

Elementul nou al acestei metode il constituie modificarea subrutinei de MAIN EXECUTION (#12A2 ... #15AE). Acest lucru a fost posibil deoarece aceasta subrutina nu este apelata direct decit din secventa de initializare si astfel a putut fi relocatata in RAM. Aceasta metoda permite dezvoltarea ulterioara a unui nou sistem de operare.

2.1. CARACTERISTICI GENERALE

Punctul de plecare al acestei versiuni a fost eliminarea redundantei data de verificarea sintactica. In acest context, interpretorul permite introducerea oricarui text BASIC, ignorind detectarea unei erori. La executie, erorile sint interceptate prin tratarea intreruperii in IM 2. Aici se realizeaza o discutie care separa erorile de extensii. Extensiile dau erori de tipul NONSENSE IN BASIC.

Totusi, in aceasta versiune este prezent un compromis. Astfel utilizatorul este avertizat in momentul in care incearca sa introduca o linie incorecta din punctul de vedere al Sinclair BASIC-ului.

Sintaxa noilor instructiuni este de forma:

<numar_linie> * <cuvint_cheie> [{parametru,}]

Avantaje:

- elimina (mai exact evita) redundanta. Verificarea sintactica se face si la introducerea "liniei tentativa" dar detectarea unei erori nu poate opri transferarea liniei in zona de program.
- compatibilitate 100 % cu interpretorul BASIC.
- scade riscul iesirii accidentale din modul de extensie.
- separarea erorilor de extensii este foarte rapida.
- quasiindependenta de perifericul atasat la EXPANSION PORT

Dezavantaje:

- in versiunea curenta exista inca limitari iar responsabilitatea introducerii corecte a noilor instructiuni revine in intregime utilizatorului.
- probabil numele noilor instructiuni nu reflecta pe deplin functiile lor (fapt intilnit si la alte extensii).
- in scopul folosirii memoriei ramase pentru programul BASIC in aplicatii mai ample, s-a renuntat la utilizarea atributelor de culoare.

2.2. DESCRIEREA METODEI

Se genereaza tabela de vectorizare a intreruperii in IM 2 si se stabileste acest mod de intrerupere (subrutina EXTON).

Saltul neconditionat la INIT determina afisarea mesajului de copyright si intrarea in propria subrutina de MAIN EXECUTION cu pozitionarea corecta a stivei si a ERRSP-ului.

Noua MAIN EXECUTION functioneaza dupa cum urmeaza : dupa verificarea sintactica, daca a fost detectata o eroare, aceasta se anuleaza iar anularea este semnalizata acustic si printr-un mesaj de avertizare.

In modul de executie, instructiunea HALT de la L1303 (atentie, eticheta, nu valoare absoluta) duce la asteptarea intreruperii.

Subrutina de tratare a intreruperii (EXST) realizeaza discutia care identifica noile instructiuni (CALL ERROR) dupa ce in prealabil a apelat tratarea standard a intreruperii de la #38.

Discutia de identificare se face in urmatoarea succesiune:

- daca intreruperea provine de la HALT-ul de la L1303, inseamna ca a fost detectata o eroare in timpul executiei si in acest caz se verifica daca este o eroare de tip NONSENSE IN BASIC.
- daca este o eroare de alt tip, tratarea ei se lasa in seama interpretorului BASIC.
- daca eroarea este de tip NONSENSE IN BASIC si primul caracter din linie este caracterul "*", se anuleaza eroarea in ERRNR si se cerceteaza lista noilor instructiuni (MYCOM). Daca se depaseste capatul listei, se genereaza o eroare de tip NONSENSE IN BASIC. Pozitionarea instructiunilor in lista a fost facuta dupa frecventa estimata a utilizarii lor.
- cind a fost identificata o noua instructiune, se apeleaza subrutina de comanda a acesteia. Aceasta realizeaza preluarea parametrilor, verificindu-le numarul, tipul si intervalul de valori permise. Daca verificarile mentionate dau gres, se genereaza mesaje de eroare corespunzatoare (tipic NONSENSE IN BASIC si INTEGER OUT OF RANGE).

Dupa executia subrutinei de comanda se trece la interpretarea liniei urmatoare (JP GOTO (7786)).

2.3. LIMITARI

Ne permitem sa amintim ca acestea sint limitari ale versiunii curente si nu limitari ale metodei. Versiuni ulterioare le vor elimina.

- parametrii unei instructiuni noi nu pot fi constante numerice. Aceasta limitare este datorata faptului ca trecerea unei linii in zona de program nu este precedata de adaugarea formei in virgula flotanta, constantelor numerice in cazul in care linia este incorecta pentru Sinclair BASIC. Se poate folosi artificiul:

const_numerica = VAL "const_numerica"

ATENTIUNE ! in expresii complicate functia VAL poate da surprize (BUG-uri din ROM).

- noile instructiuni nu pot fi urmate de alte instructiuni pe aceeasi linie. Motivul este o simplificare care face ca dupa executia unei instructiuni noi sa se treaca la interpretarea liniei urmatoare, fara a mai verifica prezenta separatorului ":".

- noile instructiuni nu pot fi date ca si comenzi deoarece nu a fost prevazuta o ramura care sa tina cont de aceasta posibilitate.

- daca se foloseste o interfata ZX 1 cu numar de serie mai mare decit 87316, ultima instructiune din program trebuie sa fie o instructiune normala (nu o extensie). Altfel, se paraseste modul de extindere intrindu-se in subrutina MAIN EXECUTION din ROM. De asemenea, orice eroare specifica interfetei ZX 1 are acelasi efect. Cauzele posibile trebuie cautate in sistemul de operare al interfetei. Remedierea acestui neajuns se face relansind modul de extensie cu RANDOMIZE USR 6E4.

3. INSTRUCTIUNI DE EXTINDERE A POSIBILITATILOR GRAFICE

In studiul curbelor de sinteza pozitionala a mecanismelor cu bare s-a facut simtita nevoia unei rezolutii grafice superioare celei oferite de SPECTRUM-ul standard (256 * 192 pixeli). Deoarece memoria alocata paginii video are o structura fixa (de la #4000 cu o lungime de 6 Kbytes excluzind atributele de culoare), a fost necesara rezervarea unei zone de patru ori mai mare incepind de la adresa #8000. In acest mod, RAMTOP-ul a fost fixat la #8000, SPECTRUM-ul comportindu-se ca o masina in configuratia cu 16 Kbytes RAM. Partea superioara a memoriei a fost rezervata subrutinelor de extindere si comanda.

Ecranul propriu-zis este folosit ca o fereastra care se comuta (automat) in zona de interes.

3.1. INSTRUCTIUNI NOI

3.1.1. * POINT <x>,<y> unde 0 <= x <= 511 si 0 <= y <= 383

Seteaza un pixel la coordonatele x,y. Sistemul de coordonate este la fel orientat cu cel al SPECTRUM-ului.

3.1.2. * CLOSE

Comutarea automata intre ecrane se face doar cind noul pixel care tebuie setat se afla in alt ecran decit cel curent. Astfel dupa ultima operatie de plotare ecranul curent trebuie "fixat" in zona corespunzaroare de memorie. Este tocmai ceea ce realizeaza aceasta instructiune.

3.1.3. * ERASE <numar_ecran> unde 1 <= numar_ecran <= 4

Sterge ecranul indicat de parametrul numar_ecran.

3.1.4. * SCREEN$ <numar_ecran> unde 1 <= numar_ecran <= 4

Afiseaza pe monitor ecranul indicat de parametrul numar_ecran.

3.1.5. * MOVE <numar_ecran> unde 1 <= numar_ecran <= 4

Copiaza continutul ecranului curent in ecranul indicat de parametrul numar_ecran. Pozitia relativa a celor patru ecrane este:

1 2
3 4

Astfel se permite introducerea in imaginea de inalta rezolutie a unor ecrane prelucrate cu un program de grafica.

3.1.6. * PAPER <scara> unde 1 <= scara <= 2

Face o copie pe imprimanta a celor patru ecrane alaturate corespunzator la scara indicata de parametrul scara. Copierea se

face pe o imprimanta SCAMP prin intermediul interfetei seriale RS 232 din interfata ZX 1. Se obtine astfel pe hirtia imprimantei o imagine de 512 * 384 pixeli la scara 1 sau 1024 * 768 pixeli la scara 2.

3.1.7. * SQR

Este o instructiune fara parametrii care are ca efect introducerea unor "coltare" ce permit prin pozitia si orientarea lor identificarea ecranului curent (ecranului in lucru). De asemenea la copierea pe imprimanta permite delimitarea imaginii pe o hirtie de format mai mare.

3.1.8. * STOP

Determina iesirea din modul de extensie (EXOFF).

3.2. EXEMPLE DE PROGRAME

Programul de mai jos traseaza graficul functiei sinus evidentiind astfel modul de functionare al extensiilor. Instructiunea * PAPER nu functioneaza decit daca este atasata interfata ZX 1. STOP-ul de la linia 150 permite raminerea in modul de extensie.

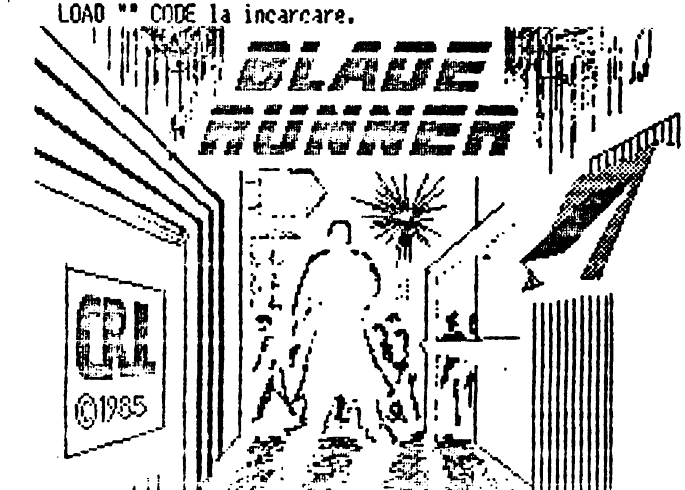
```
10 FOR I=1 TO 4
20 * ERASE I
30 NEXT I
40 * SQR
50 FOR X=0 TO 511
60 LET Y=191+191*SIN (X/511*2*PI)
70 * POINT X,Y
80 NEXT X
90 * CLOSE #
100 FOR I=1 TO 4
110 PAUSE 0
120 * SCREEN$ I
130 NEXT I
140 * PAPER VAL "1"
150 STOP
```

Programul de mai jos permite incarcarea a patru ecrane de pe banda si copierea lor in cele patru ecrane extinse urmata de copierea la imprimanta la scara 2 si iesirea din modul de extensie.

```
10 FOR I=1 TO 4
20 LOAD "" SCREEN$
30 * MOVE I
40 NEXT I
50 * PAPER VAL "2"
60 * STOP
70 STOP
```

Pentru a salva respectiv incarca pe/de pe banda toate cele patru ecrane folositi:

SAVE "nume_fisier" CODE 32768,24576 la salvare si
LOAD "" CODE la incarcare.



4. CONCLUZII

Lucrarea prezinta O NOUA METODA DE EXTINDERE A INTERPRETORULUI BASIC care aplicata posibilitatilor grafice ale computerului ZX SPECTRUM de 48 K, a condus practic la DUBLAREA REZOLUTIEI (512 * 384 pixeli).

Intentia autorilor este ca pentru computerul ZX SPECTRUM de 128 K (si compatibile) sa realizeze in mod similar o rezolutie TRIPLA (768 * 576 pixeli) sau chiar QUADRUPLA (1024 * 768 pixeli).

Se reda mai jos programul in limbaj de asamblare pentru noua metoda de extindere precum si loaderul BASIC al acestuia.

BIBLIOGRAFIE COMENTATA

[1] I. Logan, F. O'Hara, The complete SPECTRUM ROM disassembly, 1983, Melbourne House Publishers.

- o carte esentiala pentru intelegerea si dezvoltarea sistemului de operare si interpretorului BASIC. Etichetele folosite in acest articol au aceleasi nume sau sint inspirate din aceasta carte.

[2] D. Webb, Advanced Spectrum machine language, 1983, Melbourne House Publishers.

- explica intre altele "tot ceea ce trebuie sa sti despre intreruperi" intr-un Sinclair Spectrum. Generarea tabelei pentru vectorii modului 2 de intrerupere a fost inspirata din aceasta carte. In plus, cartea induce un stil foatre bun de programare in cod masina.

[3] I. Logan, SPECTRUM MICRODRIVE book with details of the ZX Interface 1; ..., 1983, Melbourne House Publishers.

- discuta in capitolele de cod masina apelarea subrutinelor din ROM-ul interfetei ZX 1 cu asa numitele HOOK CODE (coduri de eroare cu destinatie speciala) precum si posibilitatea extinderii interpretorului BASIC pe seama interfetei.

[4] S. Vickers, SPECTRUM BASIC programming, 1982, Sinclair Research Ltd.

- a fost folosita pentru prezentarea variabilelor de sistem si explicarea succinta a functiilor lor.

[5] *** Z80 CPU-instruction set, Zilog

[6] *** Manuale de utilizare pentru BLAST COMPILER, COLT COMPILER, BETA BASIC

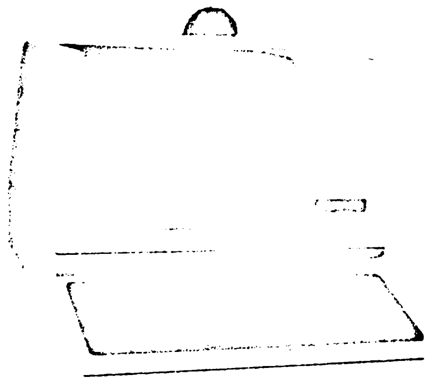
```
*HISOFT GENS3M2 ASSEMBLER*
      ZX SPECTRUM

Copyright (C) HISOFT 1983,4
All rights reserved

Pass 1 errors: 00

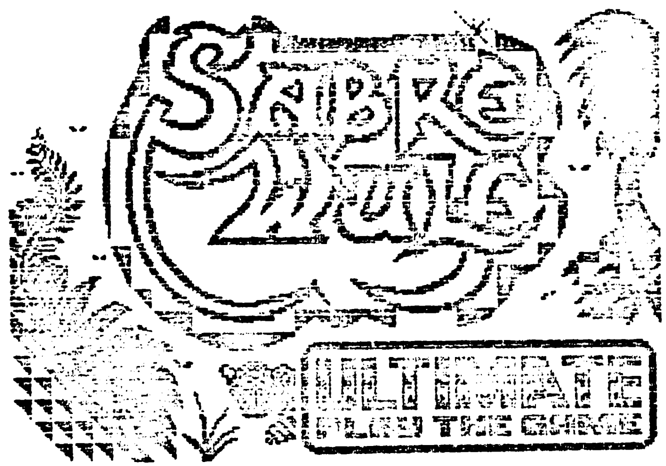
                  100 ************************
                  110 * BASIC EXTENSIONS VBS  *
                  120 ************************
03B5              130 BEEPER EQU  949
5C48              140 BORDCR EQU  23624
5C5D              150 CHRADD EQU  23645
00A0              160 ENDMSG EQU  160
5C3A              170 ERRNR  EQU  23610
5C3D              180 ERRSP  EQU  23613
2DA2              190 FP_BC  EQU  #2DA2
0018              200 GET_CH EQU  #18
1E6A              210 GOTO   EQU  7786
0038              220 INT1   EQU  #38
12A9              230 MAIN_1 EQU  #12A9
0020              240 NXT_CH EQU  #20
0C0A              250 PO_MSG EQU  #0C0A
5C45              260 PPC    EQU  23621
5CB2              270 RAMTOP EQU  23730
24FB              280 SCAN   EQU  #24FB
1800              290 SCLEN  EQU  6144
0080              300 STMSG  EQU  #80
1B76              310 STRET  EQU  7030
01FF              320 XMAX   EQU  511
017F              330 YMAX   EQU  383
                  340 ************************
FDFD              350        ORG  #FDFD
FDFD C3CDEA       360        JP   INT2
                  370 ************************
8000              380        ORG  #8000
8000              390 S1     DEFS SCLEN
9800              400 S2     DEFS SCLEN
```

```
B000                S3      DEFS SCLEN
C800                S4      DEFS SCLEN
                    ************************
EA60                        ORG  60000
EA60                        ENT  $
                    ************************
EA60 CD66EA         START   CALL EXTON
EA63 C31AEB                 JP   INIT
                    ************************
EA66 2100FE         EXTON   LD   HL,#FE00
EA69 01FD00                 LD   BC,#00FD
EA6C 71             EX      LD   (HL),C
EA6D 23                     INC  HL
EA6E 10FC                   DJNZ EX
EA70 71                     LD   (HL),C
EA71 3EFE                   LD   A,#FE
EA73 ED47                   LD   I,A
EA75 ED5E                   IM   2
EA77 FB                     EI
EA78 C9                     RET
                    ************************
EA79 3E3E           EXTOFF  LD   A,#3E
EA7B ED56                   IM   1
EA7D ED47                   LD   I,A
EA7F FB                     EI
EA80 C9                     RET
                    ************************
                    ;VARIABILE DE PROGRAM
                    ************************
EA81 0000           RETAD   DEFW 0
EA83 0000           Y       DEFW 0
EA85 0000           X       DEFW 0
EA87 00             YC      DEFB 0
EA88 00             XC      DEFB 0
EA89 00             NEWSN   DEFB 0
EA8A 00             OLDSN   DEFB 0
EA8B 00000000       BUF1    DEFB 0,0,0,0,0,0
EA91 807F           CYRMSG  DEFB STMSG,127
EA93 31393838               DEFM "1988 OMM & VBS 4SCREEN$ V1.0"
EAAF A0                     DEFB ENDMSG
EAB0 80             WRNMSG  DEFB STMSG
EAB1 5741524E               DEFM "WARNING EXTENSION ASSUMED !"
EACC A0                     DEFB ENDMSG
                    ************************
EACD E3             INT2    EX   (SP),HL
EACE 2281EA                 LD   (RETAD),HL
EAD1 E3                     EX   (SP),HL
EAD2 F5                     PUSH AF
EAD3 C5                     PUSH BC
EAD4 D5                     PUSH DE
EAD5 E5                     PUSH HL
EAD6 FF                     RST  INT1   2
EAD7 F3                     DI
EAD8 CDE1EA                 CALL ERROR
EADB E1                     POP  HL
EADC D1                     POP  DE
EADD C1                     POP  BC
EADE F1                     POP  AF
EADF FB                     EI
EAE0 C9                     RET
                    ************************
EAE1 2A81EA         ERROR   LD   HL,(RETAD)
EAE4 11AFEB                 LD   DE,L1303+1
EAE7 B7                     OR   A
EAE8 ED52                   SBC  HL,DE
EAEA C0                     RET  NZ ;NO RTERROR
                    ************************
EAEB 3A3A5C         RTERR   LD   A,(ERRNR)
EAEE FE0B                   CP   #0B ;NONSENSE IN BASIC
EAF0 C0                     RET  NZ ;TRUE ERROR
                    ************************
EAF1 2A5D5C         NIB     LD   HL,(CHRADD)
EAF4 2B                     DEC  HL
EAF5 225D5C                 LD   (CHRADD),HL
EAF8 DF                     RST  GET_CH
EAF9 FE2A                   CP   "*"
EAFB C0                     RET  NZ ;TRUE NIB ERROR
                    ************************
EAFC CD90EC         MYERR   CALL MYCOM
                    ************************
EAFF 2A3D5C                 LD   HL,(ERRSP)
EB02 F9                     LD   SP,HL
EB03 11AEEB                 LD   DE,L1303
EB06 73                     LD   (HL),E
EB07 2B                     DEC  HL
EB08 72                     LD   (HL),D
EB09 21761B                 LD   HL,STRET
EB0C E5                     PUSH HL
EB0D FD3600FF               LD   (IY+0),#FF
                    ;EROARE ANULATA
EB11 ED4B455C               LD   BC,(PPC)
EB15 03                     INC  BC
EB16 FB                     EI
EB17 C36A1E                 JP   GOTO
                    ************************
EB1A 2AB25C         INIT    LD   HL,(RAMTOP)
```



```
EB1D 363E      1370         LD   (HL),#3E
EB1F 2B        1380         DEC  HL
EB20 F9        1390         LD   SP,HL
EB21 2B        1400         DEC  HL
EB22 2B        1410         DEC  HL
EB23 223D5C    1420         LD   (ERRSP),HL
EB26 CD53EE    1430         CALL CLS1
EB29 CD4EEE    1440         CALL CLS2
EB2C CD49EE    1450         CALL CLS3
EB2F CD44EE    1460         CALL CLS4
EB32 CD7EEF    1470         CALL SQR
EB35 1191EA    1480         LD   DE,CYRMSG
EB38 AF        1490         XOR  A
EB39 CD0A0C    1500         CALL PO MSG
EB3C FDCB02EE  1510         SET  5,(IY+2);TVFL
EB40 1807      1520         JR   L12A9
EB42 FD363102  1530 L12A2   LD   (IY+49),#02
EB46 CD9517    1540         CALL #1795
EB49 CDB016    1550 L12A9   CALL #16B0
EB4C 3E00      1560 L12AC   LD   A,#00
EB4E CD0116    1570         CALL #1601
EB51 CD2C0F    1580         CALL #0F2C
EB54 CD171B    1590         CALL #1B17
EB57 FDCB007E  1600         BIT  7,(IY+0)
EB5B CCFDEC    1610         CALL Z,WARN
EB5E FD3600FF  1620         LD   (IY+0),#FF
               1630 ;EROARE ANULATA
EB62 FDCB007E  1640         BIT  7,(IY+0)
EB66 2012      1650         JR   NZ,L12CF
EB68 FDCB3066  1660         BIT  4,(IY+48)
EB6C 2840      1670         JR   Z,L1303
EB6E 2A595C    1680         LD   HL,(#5C59)
EB71 CDA711    1690         CALL #11A7
EB74 FD3600FF  1700         LD   (IY+0),#FF
EB78 18D2      1710         JR   L12AC
EB7A 2A595C    1720 L12CF   LD   HL,(#5C59)
EB7D 225D5C    1730         LD   (#5C5D),HL
EB80 CDFB19    1740         CALL #19FB
EB83 78        1750         LD   A,B
EB84 B1        1760         OR   C
EB85 C23EEC    1770         JP   NZ,L155D
EB88 DF        1780         RST  #18
EB89 FE0D      1790         CP   #0D
EB8B 28B5      1800         JR   Z,L12A2
EB8D FDCB3046  1810         BIT  0,(IY+48)
EB91 C4AF0D    1820         CALL NZ,#0DAF
EB94 CD6E0D    1830         CALL #0D6E
EB97 3E19      1840         LD   A.#19
EB99 FD964F    1850         SUB  (IY+79)
EB9C 328C5C    1860         LD   (#5C8C),A
EB9F FDCB01FE  1870         SET  7,(IY+1)
EBA3 FD3600FF  1880         LD   (IY+0),#FF
EBA7 FD360A01  1890         LD   (IY+10),#01
EBAB CD8A1B    1900         CALL #1B8A
EBAE 76        1910 L1303   HALT
EBAF FDCB01AE  1920         RES  5,(IY+1)
EBB3 FDCB304E  1930         BIT  1,(IY+48)
EBB7 C4CD0E    1940         CALL NZ,#0ECD
EBBA 3A3A5C    1950         LD   A,(#5C3A)
EBBD 3C        1960         INC  A
EBBE F5        1970 L1313   PUSH AF
EBBF 210000    1980         LD   HL,#0000
EBC2 FD7437    1990         LD   (IY+55),H
EBC5 FD7426    2000         LD   (IY+38),H
EBC8 220B5C    2010         LD   (#5C0B),HL
EBCB 210100    2020         LD   HL,#0001
EBCE 22165C    2030         LD   (#5C16),HL
EBD1 CDB016    2040         CALL #16B0
EBD4 FDCB37AE  2050         RES  5,(IY+55)
EBD8 CD6E0D    2060         CALL #0D6E
EBDB FDCB02EE  2070         SET  5,(IY+2)
EBDF F1        2080         POP  AF
EBE0 47        2090         LD   B,A
EBE1 FE0A      2100         CP   #0A
EBE3 3802      2110         JR   C,L133C
EBE5 C607      2120         ADD  A,#07
EBE7 CDEF15    2130 L133C   CALL #15EF
EBEA 3E20      2140         LD   A,#20
EBEC D7        2150         RST  #10
EBED 78        2160         LD   A,B
EBEE 119113    2170         LD   DE,#1391
EBF1 CD0A0C    2180         CALL #0C0A
EBF4 CD3B3B    2190         CALL #3B3B
EBF7 00        2200         NOP
EBF8 CD0A0C    2210         CALL #0C0A
EBFB ED4B455C  2220         LD   BC,(#5C45)
EBFF CD1B1A    2230         CALL #1A1B
EC02 3E2F      2240         LD   A,#2F
EC04 D7        2250         RST  #10
EC05 FD4E0D    2260         LD   C,(IY+13)
EC08 0600      2270         LD   B,#00
EC0A CD1B1A    2280         CALL #1A1B
EC0D CD9710    2290         CALL #1097
EC10 3A3A5C    2300         LD   A,(#5C3A)
EC13 3C        2310         INC  A
EC14 281D      2320         JR   Z.L1386
EC16 FE09      2330         CP   #09
EC18 2804      2340         JR   Z,L1373
EC1A FE15      2350         CP   #15
EC1C 2003      2360         JR   NZ,L1376
EC1E FD340D    2370 L1373   INC  (IY+13)
EC21 010300    2380 L1376   LD   BC,#0003
EC24 11705C    2390         LD   DE,#5C70
EC27 21445C    2400         LD   HL,#5C44
EC2A FDCB0A7E  2410         BIT  7,(IY+10)
EC2E 2801      2420         JR   Z,L1384
EC30 09        2430         ADD  HL,BC
EC31 EDB8      2440 L1384   LDDR
EC33 FD360AFF  2450 L1386   LD   (IY+10),#FF
EC37 FDCB019E  2460         RES  3,(IY+1)
EC3B C34CEB    2470         JP   L12AC
EC3E ED43495C  2480 L155D   LD   (#5C49),BC
EC42 2A5D5C    2490         LD   HL,(#5C5D)
EC45 EB        2500         EX   DE,HL
EC46 215515    2510         LD   HL,#1555
EC49 E5        2520         PUSH HL
EC4A 2A615C    2530         LD   HL,(#5C61)
EC4D 37        2540         SCF
EC4E ED52      2550         SBC  HL,DE
EC50 E5        2560         PUSH HL
EC51 60        2570         LD   H,B
EC52 69        2580         LD   L,C
EC53 CD6E19    2590         CALL #196E
EC56 2006      2600         JR   NZ,L157D
EC58 CDB819    2610         CALL #19B8
EC5B CDE819    2620         CALL #19E8
EC5E C1        2630 L157D   POP  BC
EC5F 79        2640         LD   A,C
EC60 3D        2650         DEC  A
EC61 B0        2660         OR   B
EC62 2828      2670         JR   Z,L15AB
EC64 C5        2680         PUSH BC
EC65 03        2690         INC  BC
EC66 03        2700         INC  BC
EC67 03        2710         INC  BC
EC68 03        2720         INC  BC
EC69 2B        2730         DEC  HL
EC6A ED5B535C  2740         LD   DE,(#5C53)
EC6E D5        2750         PUSH DE
EC6F CD5516    2760         CALL #1655
EC72 E1        2770         POP  HL
EC73 22535C    2780         LD   (#5C53),HL
EC76 C1        2790         POP  BC
EC77 C5        2800         PUSH BC
```

```
EC78 13              INC  DE
EC79 2A615C          LD   HL,(#5C61)
EC7C 2B              DEC  HL
EC7D 2B              DEC  HL
EC7E EDB8            LDDR
EC80 2A495C          LD   HL,(#5C49)
EC83 EB              EX   DE,HL
EC84 C1              POP  BC
EC85 70              LD   (HL),B
EC86 2B              DEC  HL
EC87 71              LD   (HL),C
EC88 2B              DEC  HL
EC89 73              LD   (HL),E
EC8A 2B              DEC  HL
EC8B 72              LD   (HL),D
EC8C F1       L15AB  POP  AF
EC8D C342EB          JP   L12A2
              ***********************
EC90 FB       MYCOM  EI
EC91 E7              RST  NXT_CH
EC92 FEA9            CP   #A9
EC94 CABCEC          JP   Z,POINT
EC97 FED4            CP   #D4
EC99 CA10EE          JP   Z,CLOSE
EC9C FEAA            CP   #AA
EC9E CA17EE          JP   Z,SCRNS
ECA1 FED2            CP   #D2
ECA3 CA2EEE          JP   Z,ERASE
ECA6 FEDA            CP   #DA
ECA8 CA61EE          JP   Z,PAPER
ECAB FED1            CP   #D1
ECAD CA4FEF          JP   Z,MOVE
ECB0 FEE2            CP   #E2
ECB2 CA6CEF          JP   Z,STOP
ECB5 FEBB            CP   #BB
ECB7 CA7EEF          JP   Z,SQR
ECBA CF              RST  #8
ECBB 0B              DEFB #0B
              ***********************
ECBC E7       POINT  RST  NXT_CH
ECBD CDFB24          CALL SCAN
ECC0 CDA22D          CALL FP_BC
ECC3 ED4385EA        LD   (X),BC
ECC7 E7              RST  NXT_CH
ECC8 CDFB24          CALL SCAN
ECCB CDA22D          CALL FP_BC
ECCE 217F01          LD   HL,YMAX
ECD1 B7              OR   A
ECD2 ED42            SBC  HL,BC
ECD4 2283EA          LD   (Y),HL
ECD7 CD25ED          CALL NRSC
ECDA CD78ED          CALL CORCUR
ECDD 3A89EA          LD   A,(NEWSN)
ECE0 218AEA          LD   HL,OLDSN
ECE3 BE              CP   (HL)
ECE4 280F            JR   Z,P2
ECE6 3A8AEA          LD   A,(OLDSN)
ECE9 CD96ED          CALL S_TO_M
ECEC 3A89EA          LD   A,(NEWSN)
ECEF 328AEA          LD   (OLDSN),A
ECF2 CDA3ED          CALL M_TO_S
ECF5 ED5B87EA P2     LD   DE,(YC)
ECF9 CDDAED          CALL PLOTEX
ECFC C9              RET
              *************************
ECFD 3A485C   WARN   LD   A,(BORDCR)
ED00 2F              CPL
ED01 32485C          LD   (BORDCR),A
ED04 11BCEA          LD   DE,WRNMSG
ED07 AF              XOR  A
ED08 CD0A0C          CALL PO_MSG
ED0B 21B004          LD   HL,1200
```

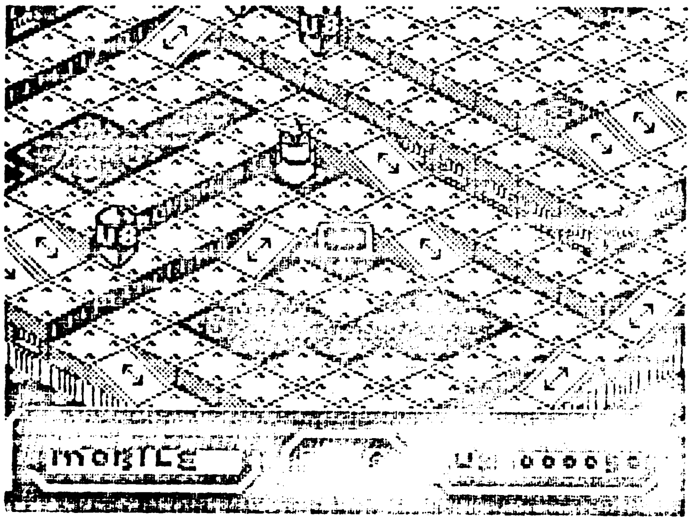



```
ED0E 112C01          LD   DE,300
ED11 CDB503          CALL BEEPER
ED14 3A485C          LD   A,(BORDCR)
ED17 2F              CPL
ED18 32485C          LD   (BORDCR),A
ED1B E638            AND  %00111000
ED1D 0F              RRCA
ED1E 0F              RRCA
ED1F 0F              RRCA
ED20 D3FE            OUT  (254),A
ED22 C9              RET
              ***********************
ED23 CF       IAOR   RST  8
ED24 0A              DEFB #0A
              ************************
ED25 3A86EA   NRSC   LD   A,(X+1)
ED28 FE01            CP   1
ED2A CA54ED          JP   Z,SC24
ED2D D223ED          JP   NC,IAOR
ED30 2A83EA   SC13   LD   HL,(Y)
ED33 118001          LD   DE,YMAX+1
ED36 B7              OR   A
ED37 ED52            SBC  HL,DE
ED39 D223ED          JP   NC,IAOR
ED3C 2A83EA          LD   HL,(Y)
ED3F 11C000          LD   DE,192
ED42 B7              OR   A
ED43 ED52            SBC  HL,DE
ED45 D24EED          JP   NC,SC3
ED48 3E01     SC1    LD   A,1
ED4A 3289EA          LD   (NEWSN),A
ED4D C9              RET
ED4E 3E03     SC3    LD   A,3
ED50 3289EA          LD   (NEWSN),A
ED53 C9              RET
ED54 2A83EA   SC24   LD   HL,(Y)
ED57 118001          LD   DE,YMAX+1
ED5A B7              OR   A
ED5B ED52            SBC  HL,DE
ED5D D223ED          JP   NC,IAOR
ED60 2A83EA          LD   HL,(Y)
ED63 11C000          LD   DE,192
ED66 B7              OR   A
ED67 ED52            SBC  HL,DE
ED69 D272ED          JP   NC,SC4
ED6C 3E02     SC2    LD   A,2
ED6E 3289EA          LD   (NEWSN),A
ED71 C9              RET
```

```
ED72 3E04     4010 SC4    LD   A,4
ED74 3289EA   4020        LD   (NEWSN),A
ED77 C9       4030        RET
              4040 ************************
ED78 3A85EA   4050 CORCUR LD   A,(X)
ED7B 3288EA   4060        LD   (XC),A
ED7E 2A83EA   4070        LD   HL,(Y)
ED81 11C000   4080        LD   DE,192
ED84 B7       4090        OR   A
ED85 ED52     4100        SBC  HL,DE
ED87 D291ED   4110        JP   NC,COCU1
ED8A 3A83EA   4120        LD   A,(Y)
ED8D 3287EA   4130        LD   (YC),A
ED90 C9       4140        RET
ED91 7D       4150 COCU1  LD   A,L
ED92 3287EA   4160        LD   (YC),A
ED95 C9       4170        RET
              4180 ************************
ED96 C5       4190 S_TO_M PUSH BC
ED97 D5       4200        PUSH DE
ED98 E5       4210        PUSH HL
ED99 CDAFED   4220        CALL PUTBAK
ED9C EDB0     4230        LDIR
ED9E E1       4240        POP  HL
ED9F D1       4250        POP  DE
EDA0 C1       4260        POP  BC
EDA1 C9       4270        RET
              4280 ************************
EDA2 C5       4290 M_TO_S PUSH BC
EDA3 D5       4300        PUSH DE
EDA4 E5       4310        PUSH HL
EDA5 CDAFED   4320        CALL PUTBAK
EDA8 EB       4330        EX   DE,HL
EDA9 EDB0     4340        LDIR
EDAB E1       4350        POP  HL
EDAC D1       4360        POP  DE
EDAD C1       4370        POP  BC
EDAE C9       4380        RET
              4390 ************************
EDAF FE01     4400 PUTBAK CP   1
EDB1 CAD0ED   4410        JP   Z,PB1
EDB4 FE02     4420        CP   2
EDB6 CACAED   4430        JP   Z,PB2
EDB9 FE03     4440        CP   3
EDBB CAC4ED   4450        JP   Z,PB3
EDBE 1100C8   4460        LD   DE,S4
EDC1 C3D3ED   4470        JP   PBK
EDC4 1100B0   4480 PB3    LD   DE,S3
EDC7 C3D3ED   4490        JP   PBK
EDCA 110098   4500 PB2    LD   DE,S2
EDCD C3D3ED   4510        JP   PBK
EDD0 110080   4520 PB1    LD   DE,S1
EDD3 210040   4530 PBK    LD   HL,16384
EDD6 010018   4540        LD   BC,SCLEN
EDD9 C9       4550        RET
              4560 ************************
EDDA F5       4570 PLOTEX PUSH AF
EDDB C5       4580        PUSH BC
EDDC D5       4590        PUSH DE
EDDD E5       4600        PUSH HL
EDDE 6A       4610        LD   L,D
EDDF CB3D     4620        SRL  L
EDE1 CB3D     4630        SRL  L
EDE3 CB3D     4640        SRL  L
EDE5 7B       4650        LD   A,E
EDE6 CB27     4660        SLA  A
EDE8 CB27     4670        SLA  A
EDEA E6E0     4680        AND  %11100000
EDEC B5       4690        OR   L
EDED 6F       4700        LD   L,A
EDEE 7B       4710        LD   A,E
EDEF CB3F     4720        SRL  A
EDF1 CB3F     4730        SRL  A
EDF3 CB3F     4740        SRL  A
EDF5 E618     4750        AND  %00011000
EDF7 F640     4760        OR   %01000000
EDF9 67       4770        LD   H,A
EDFA 7B       4780        LD   A,E
EDFB E607     4790        AND  %00000111
EDFD B4       4800        OR   H
EDFE 67       4810        LD   H,A
EDFF 7A       4820        LD   A,D
EE00 E607     4830        AND  %0000111
EE02 47       4840        LD   B,A
EE03 04       4850        INC  B
EE04 AF       4860        XOR  A
EE05 37       4870        SCF
EE06 1F       4880 PLEX1  RRA
EE07 10FD     4890        DJNZ PLEX1
EE09 B6       4900        OR   (HL)
EE0A 77       4910        LD   (HL),A
EE0B E1       4920        POP  HL
EE0C D1       4930        POP  DE
EE0D C1       4940        POP  BC
EE0E F1       4950        POP  AF
EE0F C9       4960        RET
              4970 ************************
EE10 3A89EA   4980 CLOSE  LD   A,(NEWSN)
EE13 CD96ED   4990        CALL S_TO_M
EE16 C9       5000        RET
              5010 ************************
EE17 3A89EA   5020 SCRNS  LD   A,(NEWSN)
EE1A 328AEA   5030        LD   (OLDSN),A
EE1D E7       5040        RST  NXT_CH
EE1E CDFB24   5050        CALL SCAN
EE21 CDA22D   5060        CALL FP_BC
EE24 CD5FEF   5070        CALL TESTSN
EE27 3289EA   5080        LD   (NEWSN),A
EE2A CDA2ED   5090        CALL M_TO_S
EE2D C9       5100        RET
              5110 ************************
EE2E E7       5120 ERASE  RST  NXT_CH
EE2F CDFB24   5130        CALL SCAN
EE32 CDA22D   5140        CALL FP_BC
EE35 CD5FEF   5150        CALL TESTSN
EE38 FE01     5160        CP   1
EE3A 2817     5170        JR   Z,CLS1
EE3C FE02     5180        CP   2
EE3E 280E     5190        JR   Z,CLS2
EE40 FE03     5200        CP   3
EE42 2805     5210        JR   Z,CLS3
EE44 2100C8   5220 CLS4   LD   HL,S4
EE47 180D     5230        JR   ER1
EE49 2100B0   5240 CLS3   LD   HL,S3
EE4C 1808     5250        JR   ER1
EE4E 210098   5260 CLS2   LD   HL,S2
EE51 1803     5270        JR   ER1
EE53 210080   5280 CLS1   LD   HL,S1
EE56 AF       5290 ER1    XOR  A
EE57 77       5300        LD   (HL),A
EE58 54       5310        LD   D,H
EE59 5D       5320        LD   E,L
EE5A 13       5330        INC  DE
EE5B 010018   5340        LD   BC,SCLEN
EE5E EDB0     5350        LDIR
EE60 C9       5360        RET
              5370 ************************
EE61 E7       5380 PAPER  RST  NXT_CH
EE62 CDFB24   5390        CALL SCAN
EE65 CDA22D   5400        CALL FP_BC
EE68 DA23ED   5410        JP   C,IAOR
EE6B B7       5420        OR   A
EE6C CA23ED   5430        JP   Z,IAOR
EE6F FE03     5440        CP   3
EE71 D223ED   5450        JP   NC,IAOR
EE74 FE02     5460        CP   2
```

```
EE76 CABFEF    5470          JP   Z,PAP_S2
               5480 ;
EE79 CD25EF    5490 PAP_S1   CALL IN_MG
EE7C 1E00      5500          LD   E,0
EE7E 0620      5510          LD   B,32
EE80 C5        5520 PAP1     PUSH BC
EE81 3E01      5530          LD   A,1
EE83 CDA2ED    5540          CALL M_TO_S
EE86 CDBEEE    5550          CALL TRL
EE89 3E02      5560          LD   A,2
EE8B CDA2ED    5570          CALL M_TO_S
EE8E CDBEEE    5580          CALL TRL
EE91 CD36EF    5590          CALL SND_NL
EE94 C1        5600          POP  BC
EE95 7B        5610          LD   A,E
EE96 C606      5620          ADD  A,6
EE98 5F        5630          LD   E,A
EE99 10E5      5640          DJNZ PAP1
               5650 ;
EE9B 1E00      5660          LD   E,0
EE9D 0620      5670          LD   B,32
EE9F C5        5680 PAP2     PUSH BC
EEA0 3E03      5690          LD   A,3
EEA2 CDA2ED    5700          CALL M_TO_S
EEA5 CDBEEE    5710          CALL TRL
EEA8 3E04      5720          LD   A,4
EEAA CDA2ED    5730          CALL M_TO_S
EEAD CDBEEE    5740          CALL TRL
EEB0 CD36EF    5750          CALL SND_NL
EEB3 C1        5760          POP  BC
EEB4 7B        5770          LD   A,E
EEB5 C606      5780          ADD  A,6
EEB7 5F        5790          LD   E,A
EEB8 10E5      5800          DJNZ PAP2
EEBA CD30EF    5810          CALL IES_MG
EEBD C9        5820          RET
               5830 ;
EEBE 1600      5840 TRL      LD   D,0
EEC0 0620      5850          LD   B,32
EEC2 C5        5860 TRL1     PUSH BC
EEC3 CDD1EE    5870          CALL FILBUF
EEC6 CDE4EE    5880          CALL SNDBUF
EEC9 C1        5890          POP  BC
EECA 7A        5900          LD   A,D
EECB C608      5910          ADD  A,8
EECD 57        5920          LD   D,A
EECE 10F2      5930          DJNZ TRL1
EED0 C9        5940          RET
               5950 ;
EED1 D5        5960 FILBUF   PUSH DE
EED2 218BEA    5970          LD   HL,BUF1
EED5 0606      5980          LD   B,6
EED7 E5        5990 FLB1     PUSH HL
EED8 CD01EF    6000          CALL ADR_2
EEDB 7E        6010          LD   A,(HL)
EEDC E1        6020          POP  HL
EEDD 77        6030          LD   (HL),A
EEDE 23        6040          INC  HL
EEDF 1C        6050          INC  E
EEE0 10F5      6060          DJNZ FLB1
EEE2 D1        6070          POP  DE
EEE3 C9        6080          RET
               6090 ;
EEE4 0608      6100 SNDBUF   LD   B,8
EEE6 C5        6110 SDB2     PUSH BC
EEE7 218BEA    6120          LD   HL,BUF1
EEEA 0606      6130          LD   B,6
EEEC CB16      6140 SDB1     RL   (HL)
EEEE CB19      6150          RR   C
EEF0 23        6160          INC  HL
EEF1 10F9      6170          DJNZ SDB1
EEF3 CB39      6180          SRL  C
EEF5 CB39      6190          SRL  C
EEF7 79        6200          LD   A,C
EEF8 F6C0      6210          OR   %11000000
EEFA CD3CEF    6220          CALL SND_A
EEFD C1        6230          POP  BC
EEFE 10E6      6240          DJNZ SDB2
EF00 C9        6250          RET
               6260 ;
               6270 ***********************
EF01 F5        6280 ADR_2    PUSH AF
EF02 6A        6290          LD   L,D
EF03 CB3D      6300          SRL  L
EF05 CB3D      6310          SRL  L
EF07 CB3D      6320          SRL  L
EF09 7B        6330          LD   A,E
EF0A CB27      6340          SLA  A
EF0C CB27      6350          SLA  A
EF0E E6E0      6360          AND  %11100000
EF10 B5        6370          OR   L
EF11 6F        6380          LD   L,A
EF12 7B        6390          LD   A,E
EF13 CB3F      6400          SRL  A
EF15 CB3F      6410          SRL  A
EF17 CB3F      6420          SRL  A
EF19 E618      6430          AND  %00011000
EF1B F640      6440          OR   %01000000
EF1D 67        6450          LD   H,A
EF1E 7B        6460          LD   A,E
EF1F E607      6470          AND  %00000111
EF21 B4        6480          OR   H
EF22 67        6490          LD   H,A
EF23 F1        6500          POP  AF
EF24 C9        6510          RET
               6520 ***********************
EF25 3E1B      6530 IN_MG    LD   A,#1B
EF27 CD3CEF    6540          CALL SND_A
EF2A 3E47      6550          LD   A,#47
EF2C CD3CEF    6560          CALL SND_A
EF2F C9        6570          RET
               6580 ***********************
EF30 3E2D      6590 IES_MG   LD   A,#2D
EF32 CD3CEF    6600          CALL SND_A
EF35 C9        6610          RET
               6620 ***********************
EF36 3E2F      6630 SND_NL   LD   A,#2F
EF38 CD3CEF    6640          CALL SND_A
EF3B C9        6650          RET
               6660 ***********************
EF3C F5        6670 SND_A    PUSH AF
EF3D C5        6680          PUSH BC
EF3E D5        6690          PUSH DE
EF3F E5        6700          PUSH HL
EF40 D9        6710          EXX
EF41 E5        6720          PUSH HL
EF42 D9        6730          EXX
EF43 F3        6740          DI
EF44 CF        6750          RST  8
EF45 1E        6760          DEFB #1E
EF46 FB        6770          EI
EF47 D9        6780          EXX
EF48 E1        6790          POP  HL
EF49 D9        6800          EXX
EF4A E1        6810          POP  HL
EF4B D1        6820          POP  DE
EF4C C1        6830          POP  BC
EF4D F1        6840          POP  AF
EF4E C9        6850          RET
               6860 ***********************
EF4F CD2000    6870 MOVE     CALL NXT_CH
EF52 CDFB24    6880          CALL SCAN
EF55 CDA22D    6890          CALL FP_BC
EF58 CD5FEF    6900          CALL TESTSN
EF5B CD96ED    6910          CALL S_TO_M
EF5E C9        6920          RET
               6930 ***********************
```

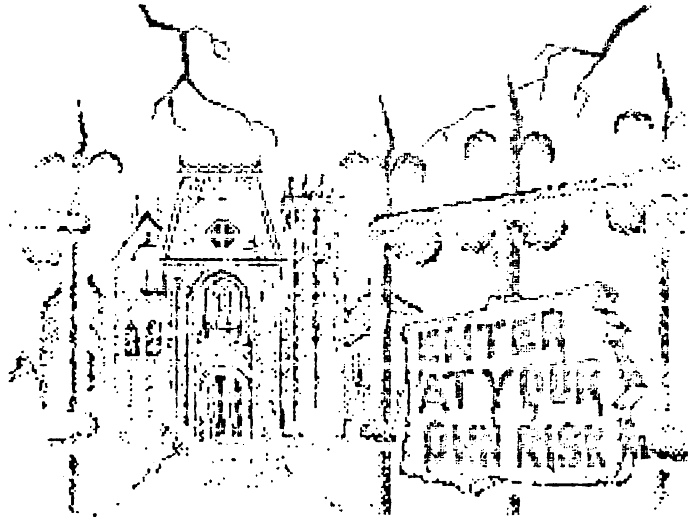
```
EF5F DA23ED   TESTSN JP   C,IADR
EF62 B7              OR   A
EF63 CA23ED          JP   Z,IADR
EF66 FE05            CP   5
EF68 D223ED          JP   NC,IADR
EF6B C9              RET
              *************************
EF6C CD79EA   STOP   CALL EXTOFF
EF6F 2AB25C          LD   HL,(RAMTOP)
EF72 363E            LD   (HL),#3E
EF74 2B              DEC  HL
```



```
EF75 F9              LD   SP,HL
EF76 2B              DEC  HL
EF77 2B              DEC  HL
EF78 223D5C          LD   (ERRSP),HL
EF7B C3A912          JP   MAIN_1
              *************************
EF7E C5       SQR    PUSH BC
EF7F E5              PUSH HL
EF80 210080          LD   HL,S1
EF83 3EFF            LD   A,255
EF85 77              LD   (HL),A
EF86 0607            LD   B,7
EF88 3E80     SQR1   LD   A,128
EF8A 24              INC  H
EF8B B6              OR   (HL)
EF8C 77              LD   (HL),A
EF8D 10F9            DJNZ SQR1
              ;
EF8F 211F98          LD   HL,S2+31
EF92 3EFF            LD   A,255
EF94 77              LD   (HL),A
EF95 0607            LD   B,7
EF97 3E01     SQR2   LD   A,1
EF99 24              INC  H
EF9A B6              OR   (HL)
EF9B 77              LD   (HL),A
EF9C 10F9            DJNZ SQR2
              ;
EF9E 21E0C7          LD   HL,S3+SCLEN-32
EFA1 3EFF            LD   A,255
EFA3 77              LD   (HL),A
EFA4 0607            LD   B,7
EFA6 3E80     SQR3   LD   A,128
EFA8 25              DEC  H
EFA9 B6              OR   (HL)
EFAA 77              LD   (HL),A
EFAB 10F9            DJNZ SQR3
              ;
EFAD 21FFDF          LD   HL,S4+SCLEN-1
EFB0 3EFF            LD   A,255
EFB2 77              LD   (HL),A
EFB3 0607            LD   B,7
EFB5 3E01     SQR4   LD   A,1
EFB7 25              DEC  H
EFB8 B6              OR   (HL)
EFB9 77              LD   (HL),A
EFBA 10F9            DJNZ SQR4
EFBC E1              POP  HL
EFBD C1              POP  BC
              ;
EFBE C9              RET
              *************************
EFBF CD25EF   PAP_S2 CALL IN_MG
EFC2 16FF            LD   D,255
              ;
EFC4 3E02     PAP2_1 LD   A,2
EFC6 CDA2ED          CALL M_TO_S
EFC9 CDB5F1          CALL TRL_T1
EFCC 3E04            LD   A,4
EFCE CDA2ED          CALL M_TO_S
EFD1 CDB5F1          CALL TRL_T1
EFD4 CD74F2          CALL SUBD3
EFD7 CD36EF          CALL SND_NL
              ;
EFDA 3E02            LD   A,2
EFDC CDA2ED          CALL M_TO_S
EFDF CDC3F1          CALL TRL_T2
EFE2 3E04            LD   A,4
EFE4 CDA2ED          CALL M_TO_S
EFE7 CDC3F1          CALL TRL_T2
EFEA CD74F2          CALL SUBD3
EFED CD36EF          CALL SND_NL
              ;
EFF0 3E02            LD   A,2
EFF2 CDA2ED          CALL M_TO_S
EFF5 CDD4F1          CALL TRL_T3
EFF8 3E04            LD   A,4
EFFA CDA2ED          CALL M_TO_S
EFFD CDD4F1          CALL TRL_T3
F000 CD74F2          CALL SUBD3
F003 CD36EF          CALL SND_NL
              ;
F006 3E02            LD   A,2
F008 CDA2ED          CALL M_TO_S
F00B CDECF1          CALL TRL_T4
F00E 3E04            LD   A,4
F010 CDA2ED          CALL M_TO_S
F013 CDECF1          CALL TRL_T4
F016 CD74F2          CALL SUBD3
F019 CD36EF          CALL SND_NL
              ;
F01C 3E02            LD   A,2
F01E CDA2ED          CALL M_TO_S
F021 CDFBF1          CALL TRL_T5
F024 3E04            LD   A,4
F026 CDA2ED          CALL M_TO_S
F029 CDFBF1          CALL TRL_T5
F02C CD74F2          CALL SUBD3
F02F CD36EF          CALL SND_NL
              ;
F032 7A              LD   A,D
F033 B7              OR   A
F034 CA7CF0          JP   Z,PAP2_5
              ;
F037 3E02            LD   A,2
F039 CDA2ED          CALL M_TO_S
F03C CD0DF2          CALL TRL_T6
F03F 3E04            LD   A,4
F041 CDA2ED          CALL M_TO_S
F044 CD0DF2          CALL TRL_T6
F047 CD74F2          CALL SUBD3
F04A CD36EF          CALL SND_NL
              ;
```

```
F04D 3E02               LD   A,2
F04F CDA2ED             CALL M_TO_S
F052 CD28F2             CALL TRL_T7
F055 3E04               LD   A,4
F057 CDA2ED             CALL M_TO_S
F05A CD28F2             CALL TRL_T7
F05D CD74F2             CALL SUBD3
F060 CD36EF             CALL SND_NL
                   ;
F063 3E02               LD   A,2
F065 CDA2ED             CALL M_TO_S
F068 CD38F2             CALL TRL_T8
F06B 3E04               LD   A,4
F06D CDA2ED             CALL M_TO_S
F070 CD38F2             CALL TRL_T8
F073 CD74F2             CALL SUBD3
F076 CD36EF             CALL SND_NL
                   ;
F079 C3C4EF             JP   PAP2_1
                   ;
F07C 06C0        PAP2_5 LD   B,192
F07E 1E00               LD   E,0
F080 3E02        PAP2_2 LD   A,2
F082 CDA2ED             CALL M_TO_S
F085 1600               LD   D,0
F087 CD01EF             CALL ADR_2
F08A 7E                 LD   A,(HL)
F08B 4F                 LD   C,A
F08C CB29               SRA  C
F08E 3E01               LD   A,1
F090 CDA2ED             CALL M_TO_S
F093 16FF               LD   D,255
F095 CD01EF             CALL ADR_2
F098 7E                 LD   A,(HL)
F099 1F                 RRA
F09A CB19               RR   C
F09C CB29               SRA  C
F09E 1F                 RRA
F09F CB19               RR   C
F0A1 CB29               SRA  C
F0A3 79                 LD   A,C
F0A4 1F                 RRA
F0A5 1F                 RRA
F0A6 F6C0               OR   %11000000
F0A8 CD79F2             CALL SND_AA
F0AB 1C                 INC  E
F0AC 10D2               DJNZ PAP2_2
F0AE 06C0               LD   B,192
F0B0 1E00               LD   E,0
F0B2 3E04        PAP2_3 LD   A,4
F0B4 CDA2ED             CALL M_TO_S
F0B7 1600               LD   D,0
F0B9 CD01EF             CALL ADR_2
F0BC 7E                 LD   A,(HL)
F0BD 4F                 LD   C,A
F0BE CB29               SRA  C
F0C0 3E03               LD   A,3
F0C2 CDA2ED             CALL M_TO_S
F0C5 16FF               LD   D,255
F0C7 CD01EF             CALL ADR_2
F0CA 7E                 LD   A,(HL)
F0CB 1F                 RRA
F0CC CB19               RR   C
F0CE CB29               SRA  C
F0D0 1F                 RRA
F0D1 CB19               RR   C
F0D3 CB29               SRA  C
F0D5 79                 LD   A,C
F0D6 1F                 RRA
F0D7 1F                 RRA
F0D8 F6C0               OR   %11000000
F0DA CD79F2             CALL SND_AA
F0DD 1C                 INC  E
F0DE 10D2               DJNZ PAP2_3
F0E0 CD36EF             CALL SND_NL
                   ;
F0E3 16FD               LD   D,253
F0E5 3E01        PAP2_4 LD   A,1
F0E7 CDA2ED             CALL M_TO_S
F0EA CD28F2             CALL TRL_T7
F0ED 3E03               LD   A,3
F0EF CDA2ED             CALL M_TO_S
F0F2 CD28F2             CALL TRL_T7
F0F5 CD74F2             CALL SUBD3
F0F8 CD36EF             CALL SND_NL
                   ;
F0FB 3E01               LD   A,1
F0FD CDA2ED             CALL M_TO_S
F100 CD38F2             CALL TRL_T8
F103 3E03               LD   A,3
F105 CDA2ED             CALL M_TO_S
F108 CD38F2             CALL TRL_T8
F10B CD74F2             CALL SUBD3
F10E CD36EF             CALL SND_NL
                   ;
F111 3E01               LD   A,1
F113 CDA2ED             CALL M_TO_S
F116 CDB5F1             CALL TRL_T1
F119 3E03               LD   A,3
F11B CDA2ED             CALL M_TO_S
F11E CDB5F1             CALL TRL_T1
F121 CD74F2             CALL SUBD3
F124 CD36EF             CALL SND_NL
                   ;
F127 3E01               LD   A,1
F129 CDA2ED             CALL M_TO_S
F12C CDC3F1             CALL TRL_T2
F12F 3E03               LD   A,3
F131 CDA2ED             CALL M_TO_S
F134 CDC3F1             CALL TRL_T2
F137 CD74F2             CALL SUBD3
F13A CD36EF             CALL SND_NL
                   ;
F13D 7A                 LD   A,D
F13E FE01               CP   1
F140 CA9EF1             JP   Z,PAP2_6
                   ;
F143 3E01               LD   A,1
F145 CDA2ED             CALL M_TO_S
F148 CDD4F1             CALL TRL_T3
F14B 3E03               LD   A,3
F14D CDA2ED             CALL M_TO_S
F150 CDD4F1             CALL TRL_T3
F153 CD74F2             CALL SUBD3
F156 CD36EF             CALL SND_NL
                   ;
F159 3E01               LD   A,1
F15B CDA2ED             CALL M_TO_S
F15E CDECF1             CALL TRL_T4
F161 3E03               LD   A,3
F163 CDA2ED             CALL M_TO_S
F166 CDECF1             CALL TRL_T4
F169 CD74F2             CALL SUBD3
F16C CD36EF             CALL SND_NL
                   ;
F16F 3E01               LD   A,1
F171 CDA2ED             CALL M_TO_S
F174 CDFBF1             CALL TRL_T5
F177 3E03               LD   A,3
F179 CDA2ED             CALL M_TO_S
F17C CDFBF1             CALL TRL_T5
F17F CD74F2             CALL SUBD3
F182 CD36EF             CALL SND_NL
                   ;
```

```
F185 3E01          LD   A,1
F187 CDA2ED        CALL M_TO_S
F18A CD0DF2        CALL TRL_T6
F18D 3E03          LD   A,3
F18F CDA2ED        CALL M_TO_S
F192 CD0DF2        CALL TRL_T6
F195 CD74F2        CALL SUBD3
F198 CD36EF        CALL SND_NL
F19B C3E5F0        JP   PAP2_4
     ;
F19E 3E01   PAP2_6 LD   A,1
F1A0 CDA2ED        CALL M_TO_S
F1A3 CD49F2        CALL TRL_TF
F1A6 3E03          LD   A,3
F1A8 CDA2ED        CALL M_TO_S
F1AB CD49F2        CALL TRL_TF
F1AE CD36EF        CALL SND_NL
F1B1 CD30EF        CALL TES_MG
F1B4 C9            RET
     ;
F1B5 06C0   TRL_T1 LD   B,192
F1B7 1E00          LD   E,0
F1B9 CD01EF TR1    CALL ADR_2
F1BC 7E            LD   A,(HL)
F1BD CD5BF2        CALL DOUBLE
F1C0 10F7          DJNZ TR1
F1C2 C9            RET
     ;
F1C3 06C0   TRL_T2 LD   B,192
F1C5 1E00          LD   E,0
F1C7 CD01EF TR2    CALL ADR_2
F1CA 7E            LD   A,(HL)
F1CB 0F            RRCA
F1CC 0F            RRCA
F1CD 0F            RRCA
F1CE CD5BF2        CALL DOUBLE
F1D1 10F4          DJNZ TR2
F1D3 C9            RET
     ;
F1D4 06C0   TRL_T3 LD   B,192
F1D6 1E00          LD   E,0
F1D8 CD01EF TR3    CALL ADR_2
F1DB 7E            LD   A,(HL)
F1DC 4F            LD   C,A
F1DD 2B            DEC  HL
F1DE 7E            LD   A,(HL)
F1DF 1F            RRA
F1E0 CB19          RR   C
```



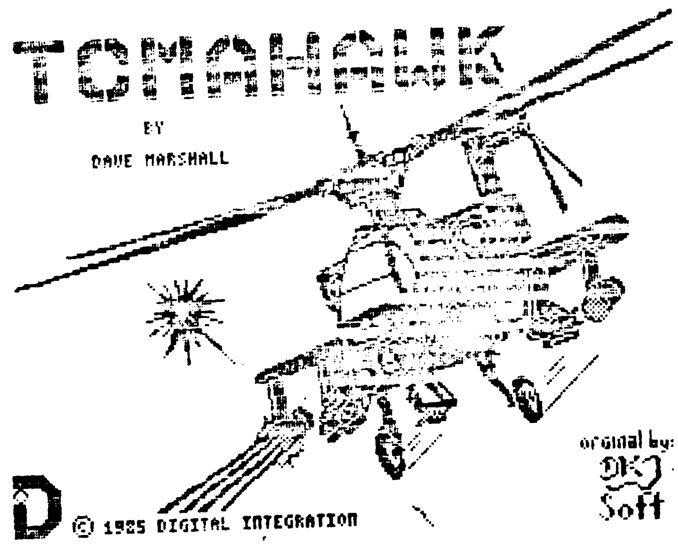

```
F1E2 79            LD   A,C
F1E3 07            RLCA
F1E4 07            RLCA
F1E5 07            RLCA
F1E6 CD5BF2        CALL DOUBLE
F1E9 10ED          DJNZ TR3
F1EB C9            RET
     ;
F1EC 06C0   TRL_T4 LD   B,192
F1EE 1E00          LD   E,0
F1F0 CD01EF TR4    CALL ADR_2
F1F3 7E            LD   A,(HL)
F1F4 1F            RRA
F1F5 CD5BF2        CALL DOUBLE
F1F8 10F6          DJNZ TR4
F1FA C9            RET
     ;
F1FB 06C0   TRL_T5 LD   B,192
F1FD 1E00          LD   E,0
F1FF CD01EF TR5    CALL ADR_2
F202 7E            LD   A,(HL)
F203 0F            RRCA
F204 0F            RRCA
F205 0F            RRCA
F206 0F            RRCA
F207 CD5BF2        CALL DOUBLE
```

```
F20A 10F3          DJNZ TR5
F20C C9            RET
     ;
F20D 06C0   TRL_T6 LD   B,192
F20F 1E00          LD   E,0
F211 CD01EF TR6    CALL ADR_2
F214 7E            LD   A,(HL)
F215 4F            LD   C,A
F216 2B            DEC  HL
F217 7E            LD   A,(HL)
F218 1F            RRA
F219 CB19          RR   C
F21B 1F            RRA
F21C CB19          RR   C
F21E 79            LD   A,C
F21F 07            RLCA
F220 07            RLCA
F221 07            RLCA
F222 CD5BF2        CALL DOUBLE
F225 10EA          DJNZ TR6
F227 C9            RET
     ;
F228 06C0   TRL_T7 LD   B,192
F22A 1E00          LD   E,0
F22C CD01EF TR7    CALL ADR_2
F22F 7E            LD   A,(HL)
F230 0F            RRCA
F231 0F            RRCA
F232 CD5BF2        CALL DOUBLE
F235 10F5          DJNZ TR7
F237 C9            RET
     ;
F238 06C0   TRL_T8 LD   B,192
F23A 1E00          LD   E,0
F23C CD01EF TR8    CALL ADR_2
F23F 7E            LD   A,(HL)
F240 07            RLCA
F241 07            RLCA
F242 07            RLCA
F243 CD5BF2        CALL DOUBLE
F246 10F4          DJNZ TR8
F248 C9            RET
     ;
F249 06C0   TRL_TF LD   B,192
F24B 1E00          LD   E,0
F24D CD01EF TRF    CALL ADR_2
F250 7E            LD   A,(HL)
F251 07            RLCA
```

```
F252 07                 RLCA
F253 E603               AND  %00000011
F255 CD5BF2             CALL DOUBLE
F258 10F3               DJNZ TRF
F25A C9                 RET
                ;
F25B 1F         DOUBLE  RRA
F25C CB19               RR   C
F25E CB29               SRA  C
F260 1F                 RRA
F261 CB19               RR   C
F263 CB29               SRA  C
F265 1F                 RRA
F266 CB19               RR   C
F268 CB29               SRA  C
F26A 79                 LD   A,C
F26B 1F                 RRA
F26C 1F                 RRA
F26D F6C0               OR   %11000000
F26F CD79F2             CALL SND_AA
F272 1C                 INC  E
F273 C9                 RET
                ;
F274 7A         SUBD3   LD   A,D
F275 D603               SUB  3
F277 57                 LD   D,A
F278 C9                 RET
                ;
F279 CD3CEF     SND_AA  CALL SND_A
F27C CD3CEF             CALL SND_A
F27F C9                 RET
                ***********************
```

Pass 2 errors: 00

```
ADR_2   EF01    BEEPER  03B5
BORDCR  5C48    BUF1    EA8B
CHRADD  5C5D    CLOSE   EE10
CLS1    EE53    CLS2    EE4E
CLS3    EE49    CLS4    EE44
COCUI   ED91    CORCUR  ED78
CYRMSG  EA91    DOUBLE  F25B
ENDMSG  00A0    ER1     EE56
ERASE   EE2E    ERRNR   5C3A
ERROR   EAE1    ERRSP   5C3D
EX      EA6C    EXTOFF  EA79
EXTON   EA66    FILBUF  EED1
FLB1    EED7    FP_BC   2DA2
GET_CH  0018    GOTO    1E6A
IACR    ED23    IES_MG  EF30
INIT    EB1A    INT1    0038
INT2    EACD    IN_MG   EF25
L12A2   EB42    L12A9   EB49
L12AC   EB4C    L12CF   EB7A
L1303   EBAE    L1313   EBBE
L133C   EBE7    L1373   EC1E
L1376   EC21    L1384   EC31
L1386   EC33    L155D   EC3E
L1570   EC5E    L15AB   EC8C
MAIN_1  12A9    MOVE    EF4F
MYCOR   EC90    MYERR   EAFC
```

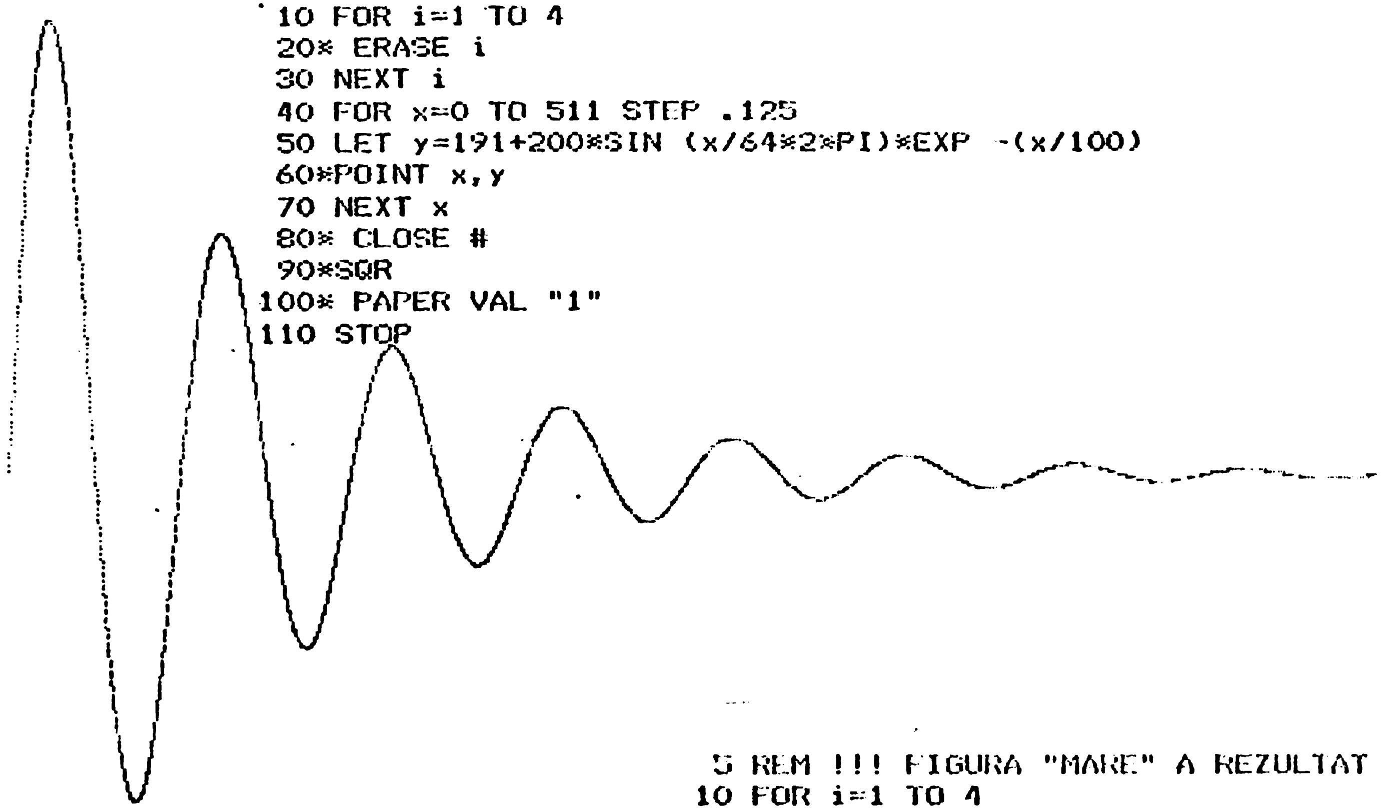


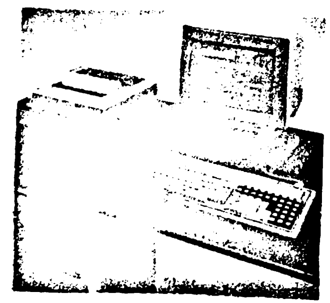

```
M_TO_S  EDA2    NEWSN   EA89
NIB     EAF1    NRSC    ED25
NXT_CH  0020    OLDSN   EA8A
P2      ECF5    PAP1    EE80
PAP2    EE9F    PAP2_1  EFC4
PAP2_2  F030    PAP2_3  F0B2
PAP2_4  F0E5    PAP2_5  F07C
PAP2_6  F19E    PAPER   EE61
PAP_S1  EE79    PAP_S2  EFBF
PB1     EDD0    PB2     EDCA
PB3     EDC4    PBK     EDD3
PLEX1   EE06    PLOTEX  EDDA
POINT   ECBC    PO_MSG  0C0A
PPC     5C45    PUTBAK  EDAF
RAMTOP  5CB2    RETAD   EA31
RTERR   EAEB    S1      8000
S2      9800    S3      B000
S4      C800    SC1     ED48
SC13    ED30    SC2     ED6C
SC24    ED54    SC3     ED4E
SC4     ED72    SCAN    24FB
SCLEN   1800    SCRNS   EE17
SDB1    EEEC    SDB2    EEE6
SNDBUF  EEE4    SND_A   EF3C
SND_AA  F279    SND_NL  EF36
SQR     EF7E    SQR1    EF88
SQR2    EF97    SQR3    EFA6
SQR4    EFB5    START   EA60
STMSG   0080    STOP    EF6C
STRET   1B76    SUBD3   F274
S_TO_M  ED96    TESTSN  EF5F
TR1     F1B9    TR2     F1C7
TR3     F1D8    TR4     F1F0
TR5     F1FF    TR6     F211
TR7     F22C    TR8     F23C
TRF     F24D    TRL     EEBE
TRL1    EEC2    TRL_T1  F1B5
TRL_T2  F1C3    TRL_T3  F1D4
TRL_T4  F1EC    TRL_T5  F1FB
TRL_T6  F20D    TRL_T7  F228
TRL_T8  F233    TRL_TF  F249
WARN    ECFD    WRNMSG  EAB0
X       EA85    XC      EA88
XMAX    01FF    Y       EA83
YC      EA87    YMAX    017F
```

Table used: 1653 from 1894
Executes: 60000

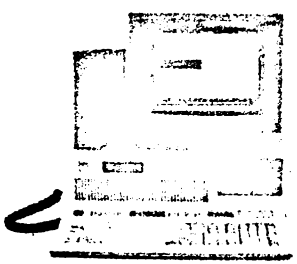
```
10 BORDER 0: PAPER 0: INK 7: CLEAR 32767
20 LOAD ""CODE : CLS : RANDOMIZE USR 6e4
```

```
  5 REM !!! FIGURA "mica" A REZULTAT DIN ACEST PROGRAM !!!
 10 FOR i=1 TO 4
 20* ERASE i
 30 NEXT i
 40 FOR x=0 TO 511 STEP .125
 50 LET y=191+200*SIN (x/64*2*PI)*EXP -(x/100)
 60*POINT x,y
 70 NEXT x
 80* CLOSE #
 90*SQR
100* PAPER VAL "1"
110 STOP
```





```
  5 REM !!! FIGURA "MARE" A REZULTAT DIN ACEST PROGRAM !!!
 10 FOR i=1 TO 4
 20* ERASE i
 30 NEXT i
 40 FOR x=0 TO 511 STEP .125
 50 LET y=191+200*SIN (x/64*2*PI)*EXP -(x/100)
 60*POINT x,y
 70 NEXT x
 80* CLOSE #
 90*SQR
100* PAPER VAL "2"
110 STOP
```



ING. TIBERU ONU

COMPRESOR DE ECRAN PENTRU CALCULATOARE SPECTRUM COMPATIBILE

1. INTRODUCERE

Se cunoaste faptul ca pagina video ocupa in calculatoarele SPECTRUM - compatibile primii 6912 octeti RAM , incepind la adresa 16384 si incheindu-se la adresa 23295. Din acesti 6912 octeti , 6144 retin configuratia de pixeli , cei 768 octeti ramasi fiind rezervati atributelor de culoare.

Pozitionind RAMTOP-ul la adresa 29999 printr-o instructiune CLEAR , in cei 35536 de octeti din zona superioara a memoriei se pot retine 5 ecrane , bineinteles in detrimentul unui program pilot relativ scurt. Aceste ecrane pot fi eventual transferate in pagina video prin intermediul unui program in cod masina care uzeaza de instructiunea LDIR.

Numarul ecranelor stocate in memorie poate creste simtitor , daca acestea se retin intr-o forma comprimata , urmind ca transferul in pagina video sa fie realizat de o rutina de decomprimare , care sa reconstituie ecranul in forma sa initiala.

In cele ce urmeaza , se prezinta principiile care pot sta la baza comprimarii ecranelor si doua subrutine care realizeaza comprimarea , respectiv decomprimarea ecranelor.

2. CONSIDERENTE LEGATE DE MEMORIA VIDEO

Primii 6144 de octeti ai memoriei video definesc starea celor 192 de linii , respectiv 256 de coloane de pixeli , de care dispune calculatorul in modul de inalta rezolutie grafica.

Fiecare din cei 256 de pixeli care formeaza o linie a ecranului , corespunde cite unui bit al memoriei video , existind o corespondenta biunivoca intre starea pixelului respectiv si starea bitului sau echivalent (bit setat <-> pixel de culoarea INK ; bit resetat <-> pixel de culoarea PAPER , corespunzator atributelor actuale de culoare). Rezulta ca o linie de pixeli va fi descrisa de un numar de 32 de octeti ai zonei video.

Referitor la organizarea zonei video , se remarca o corespondenta 'nenaturala' intre pozitia pixelilor si adresele la care sint stocati bitii care le definesc starea.

Rulind programul :

```
10 FOR x=16384 TO 22527
20 POKE x,INT (RND *256)
30 NEXT x
40 PAUSE 0
```

se observa ca , desi parcurgerea zonei video se realizeaza in ordinea fireasca a locatiilor , ecranul nu este construit din linii succesive. Se distinge o virtuala impartire a ecranului in

3 parti egale ; intii este construita integral prima treime in modul urmator : primul rind completat este rindul 0 , urmeaza rindurile 8,16,... samd pina la limita inferioara a primei treimi. Se continua cu rindurile 1,9,17 samd pina la definirea integrala a primei treimi. In continuare se construiesc in mod analog a doua si in cele din urma a ultima treime. Avantajele acestei dispuneri fizice a ecranului (tiparire rapida la nivel alfanumeric) nu vor fi tratate in acest articol.

Dupa cum se va vedea insa in cele ce urmeaza , din punctul de vedere al compresiei , in majoritatea cazurilor ar conveni o dispunere 'naturala' a ecranului de comprimat , dispunere care ar putea fi realizata dupa una din urmatoarele metode :

a) METODA ORIZONTALA. Cei 6144 de octeti se retin in locatii succesive de memorie dupa urmatorul model : intii se retin cei 32 de octeti corespunzatori primei linii de pixeli , apoi cei 32 de octeti corespunzatori celei de a doua linii de pixeli , samd pina la limita inferioara a ecranului.

b) METODA VERTICALA. Cei 6144 de octeti se retin in locatii succesive de memorie , astfel : intii se retin cei 192 de octeti corespunzatori primelor 8 coloane de pixeli , apoi se retin cei 192 de octeti corespunzatori urmatoarelor 8 coloane de pixeli , samd pina la limita din dreapta a ecranului.

Cele doua metode de dispunere pot fi aplicate si fisierului de atribute de culoare , cu specificatia ca dispunerea de facto a acestora este cea naturala orizontala.

In ambele cazuri putem afirma ca privim intregul ecran ca pe o fereastra.

3. POSIBILE MODALITATI DE COMPRIMARE A ECRANELOR

a) COMPRIMARE PARTICULARA

Aceasta modalitate de comprimare a ecranelor deriva dintr-un procedeu general de comprimare a memoriei. Aplicarea acestei metode la fisierul video al calculatoarelor de tip SPECTRUM a fost pentru prima data realizata de catre firma PRINT'N'PLOTTER , in cadrul programului SCREEN MACHINE.

Metoda se bazeaza pe faptul ca intr-un ecran , de obicei , exista multi octeti de valori 0 si 255. Comprimarea se realizeaza retinind dupa fiecare octet de valoare 0 sau 255 , ordinul sau de multiplicitate (numarul de octeti consecutivi identici).

Avind , spre exemplificare , o succesiune de 25 de octeti de forma :

255,17,17,17,1,1,1,1,1,1,8,8,37,37,207,41,23,9,0,0,0,0,0,54,101

prin comprimare particulara se ajunge la urmatorul sir de valori :

255,1,17,17,17,1,1,1,1,1,1,8,8,37,37,207,41,23,9,0,5,54,101 ,

realizindu-se un cistig de 2 octeti , respectiv 8 % din totalul initial al octetilor.

Acest tip de comprimare este insa o arma cu doua taisuri ,deoarece in cazul unui ecran cu multe valori 0 sau 255 consecutiv singulare , se poate obtine un fisier prelucrat mai lung decit cel initial ,nerealizindu-se deci o comprimare , ci o extindere a fisierului ecran.

b) COMPRIMARE GENERALA

Comprimarea generala a ecranelor se realizeaza dupa principii asemanatoare comprimarii particulare , cu singura deosebire ca ordinul multiplicitate se inscrie dupa fiecare octet , indiferent de valoarea sa.

Considerind secventa exemplificata anterior , aceasta conduce prin comprimare generala la sirul de valori :

255,1,17,3,1,6,8,2,37,2,207,1,41,1,23,1,9,1,0,5,54,1,101,1 ,

realizindu-se un cistig de 1 octet , respectiv 4 % din numarul initial de octeti.

Si in acest caz se poate obtine uneori un fisier mai lung decit cel initial , chiar mai lesne decit in cazul anterior , comprimarea generala fiind afectata de toata gama valorilor singulare , nu numai de 0 si 255.

c) COMPRIMARE GENERALA CU COD DE ESCAPE

Obtinerea unui fisier mai lung decit cel initial ar putea fi evitata in ideea folosirii unui octet specific ca si cod de escape.

In acest caz , prezenta codului de escape ar indica faptul ca dupa el urmeaza un ordin de multiplicitate. In cazul unor valori consecutiv singulare sau duble , codul de escape nu ar fi folosit (secventa : octet-cod de escape-ordin de multiplicitate este constituita din 3 octeti). In cazul unor valori consecutiv triple nu s-ar obtine practic nici o comprimare , iar in cazul oricaror valori cu ordin de multiplicitate mai mare decit trei comprimarea ar fi eficienta.

Considerind secventa data ca exemplu la punctul a) , aceasta ar conduce prin comprimarea cu un cod de escape CE , la sirul echivalent de valori :

255,17,CE,3,1,CE,6,8,8,37,37,207,41,23,9,0,CE,5,54,101 ,

realizindu-se un cistig de 5 octeti , adica 20 % din numarul i-
nitial de octeti.

Se subliniaza ideea ca CE trebuie sa fie un octet specific.El nu trebuie sa fie prezent in zona video , o situatie contrara putind da nastere la ambiguitati (nu s-ar putea discerne daca octetul cu pricina functioneaza ca si cod de escape sau pur si simplu este un octet din configuratia ecranului).

Experienta arata ca in majoritatea covirsitoare a ecranelor , indiferent de complexitatea lor , exista cel putin un octet complet nefolosit (autorul nu a intilnit inca un ecran cu rol estetic sau functional care sa contina toate valorile din plaja 0 - 255 ,desi este posibil ca astfel de ecrane sa existe). Odata gasit , un astfel de octet poate fi folosit ca si cod de escape pentru fisierul comprimat , fara a risca ambiguitati.

d) COMPRIMAREA IN FEREASTRA

In toate cele trei cazuri prezentate anterior, comprimarea obtinuta este cu mult mai drastica daca se retine ecranul nu in forma sa normala , ci sub forma de fereastra orizontala sau verticala (dupa cum s-a aratat la paragraful 2) , deoarece din considerente estetice care stau la baza constructiei oricarui ecran , in aceste cazuri creste simtitor probabilitatea de a depista octeti cu ordin mare de multiplicitate.

4. PROGRAM PENTRU COMPRIMAREA SI DECOMPRIMAREA ECRANELOR FUNCTIONAL PE CALCULATOARE SPECTRUM-COMPATIBILE

a) DESCRIERE

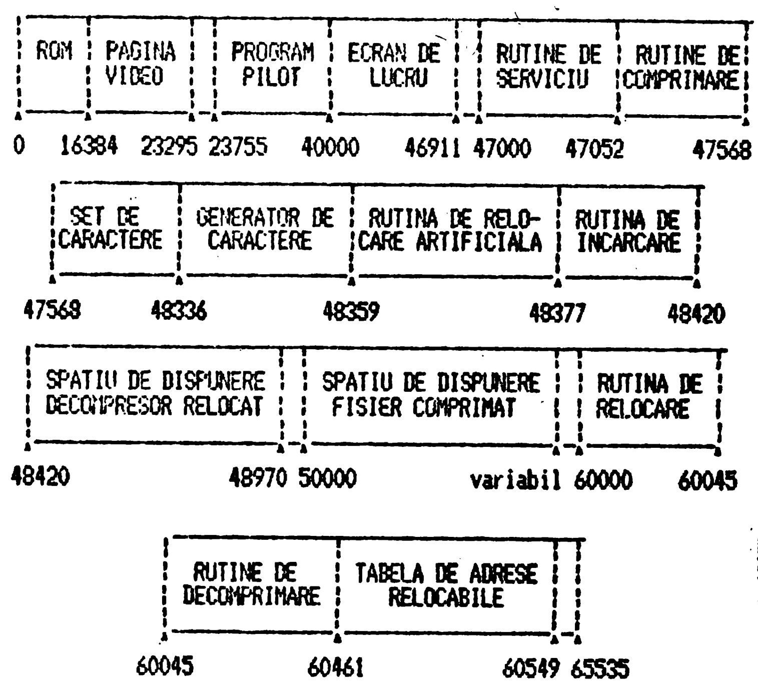
Programul exemplificat pentru comprimarea si decomprimarea ecranelor este compus din trei sectiuni principale :

PROGRAMUL PILOT - realizat in limbaj BASIC , este comandat prin meniuri si asigura usurinta in utilizare. Prin intermediul acestui program se poate comanda tipul dorit de comprimare dupa efectuarea in prealabil a unor teste legate de lungimea fisierelor comcomprimate , se poate verifica corectitudinea comprimarii si efectul decomprimarii si se pot salva fisierele comprimate in vederea utilizarii lor ulterioare in programe ale utilizatorului , laolalta cu rutina de decomprimare.

RUTINELE DE COMPRIMARE - sint realizate in limbaj de asamblare si asigura 6 tipuri de comprimare : particulara sau generala cu cod de escape , fiecare dintre acestea putind fi aplicata in configuratie normala a ecranului sau in fereastra orizontala sau verticala Rutinele de decomprimare sint apelate din programul pilot.

RUTINA DE DECOMPRIMARE - este generata intr-o forma relocabila (poate functiona in orice zona din RAM , mai putin pagina video). Recunoaste tipul de comprimare utilizat si restaureaza ecranul initial.

Harta memoriei , dupa instalarea celor trei sectiuni se prezinta astfel :



b) INSTALARE

Pentru instalarea programului se reseteaza calculatorul si se introduce intii programul pilot (listingul 1). Dupa introducere , acesta se salveaza pe caseta cu :

```
SAVE CHR$ 22+CHR$ 1 +CHR$ 0+"COM.BAS" LINE 0
```

Dupa aceasta operatie este bine sa se execute o verificare.

Se reseteaza calculatorul si se incarca un program asamblor. Poate fi folosit orice asamblor (listingul de fata a fost obtinut in LASER GENIUS) , rezultatul va fi acelasi. Se insereaza fisierul sursa din listingul 2 (rutinele de comprimare).Este indicat ca dupa inserare sa se salveze pe o alta caseta fisierul sursa. Se asambleaza programul si (eventual) se revine in BASIC de unde se salveaza pe caseta cu programul pilot , dupa acesta , codul obiect cu :

```
SAVE "COMP.OBJ" CODE adresa de start (aici 25500),1970
```

Se revine in asamblor si se sterge fisierul sursa anterior (eventual cu un start rece). Se insereaza listingul 3 (rutina de decomprimare) si se asambleaza. ATENTIUNE ! Listingul 3 incepe cu ORG 0 , in vederea relocarii , asa ca trebuie folosita o optiune de asamblare cu dispunerea deplasata a codului obiect. Se salveaza din nou codul obiect obtinut cu :

```
SAVE "DECOMP.OBJ" CODE adresa de start (aici 25500),550
```

in continuarea celor precedente.

Daca cele trei portiuni au fost corect inserate si pentru asamblare au fost respectate indicatiile de mai sus , pe caseta se obtine varianta finala a programului , care in aceste conditii este functional.

Calculatorul poate fi acum din nou resetat , iar programul poate fi incarcat in vederea utilizarii.

c) UTILIZARE

Odata incarcat programul , calculatorul pune la dispozitie urmatoarea lista de optiuni :

1. UNITATE DE INCARCARE/SALVARE - in cadrul acestei optiuni se afiseaza un submeniu care ne permite :
   - SALVAREA PROGRAMULUI COMPRESOR in vederea executarii de copii.
   - SALVAREA RUTINEI DE DECOMPRIMARE pentru utilizarea ei viitoare in programe ale utilizatorului. Rutina de decomprimare este salvata intr-o forma relocabila.
   - INCARCAREA UNUI NOU ECRAN DE LUCRU ,ecran care poate fi comprimat. Incarcarea este fara header , iar prezenta headerului este ignorata. De asemenea , nu influenteaza valoarea octetului de flag. Erorile de incarcare nu sint semnalate , fapt care da posibilitatea incarcarii unor ecrane care fac parte din blocuri cu lungimea mai mare de 6912 octeti. Incarcarea poate fi oricind abandonata prin apasare pe tasta SPACE (fara CAPS SHIFT).

2. COMPRIMARE NORMALA
3. COMPRIMARE ORIZONTALA
4. COMPRIMARE VERTICALA - aceste trei optiuni realizeaza comprimarea intr-unul din aceste formate , tipul comprimarii fiind selectat prin optiunea 6. Dupa comprimare poate fi salvat pe banda fisierul comprimat.

5. TEST - prin intermediul acestei optiuni se calculeaza lungimile fisierelor comprimate pentru fiecare din cele sase tipuri de comprimare si se afiseaza intr-un tabel in ordinea crescatoare a marimilor. De asemenea , se dau indicatii referitoare la interceptarea unui octet care ar putea fi folosit ca si cod de escape.

6. COMPRIMARE GENERALA/PARTICULARA - selecteaza tipul de comprimare care urmeaza a fi folosit de optiunile 2,3 si 4.

7. DECOMPRIMARE - realizeaza decomprimarea ultimului fisier comprimat , fara nici un efect vizibil daca anterior nu a fost comprimat nici un fisier.

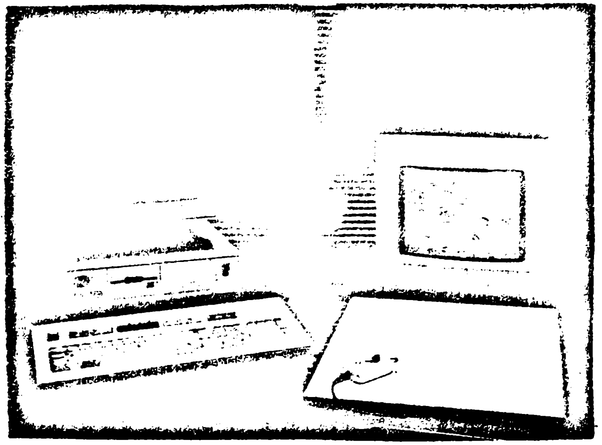
Pe tot timpul afisarii meniului principal , apasarea tastei SPACE provoaca vizualizarea ecranului de lucru.

5. CONCLUZII

Urmarind listingurile 2 si 3 , cunoscatorii limbajului de asamblare pot observa ca apelind in mod corespunzator rutinele de comprimare si decomprimare pot fi obtinute din afara programului pilot si facilitati suplimentare. O idee in acest sens ar fi de exemplu realizarea comprimarii doar pentru zona de pixeli (in cazul ecranelor monocrom sau in cazul in care se doreste apari-

tia brusca a imaginii ,in acest din urma caz atributele de culoare fiind retinute intr-o forma necomprimata si mutate ulterior printr-o instructiune LDIR). Se remarca de asemenea posibilitatea ca in cazul comprimarii in fereastra , prin modificari la adresele etichetate xx1+1,xx2+1,xx3+1 si xx4+1 sa se realizeze comprimarea si decomprimarea in ferestre mai mici decit intregul ecran.

Programul a fost testat pe o varietate larga de ecrane si a dat in toate cazurile rezultate satisfacatoare. In toate cazurile intilnite , fisierele comprimate generate rezulta mai scurte



decit cele furnizate de SCREEN MACHINE.

Conceptia programului este originala iar ideile legate de comprimarea generala cu cod de escape si comprimarea in fereastra , apartin integral lui ing. Miodrag Puterity si autorului , ele nemaifiind intilnite la programele unor firme din strainatate.

Sint luate in vedere si posibilitati de extindere a programului prezentat dintre care se amintesc : cautarea mai multor coduri de escape si utilizarea lor pentru a specifica structuri de repetitie cu lungimea mai mare de un octet sau baleierea ecranului in vederea gasirii ferestrei cu densitatea maxima de informatie si descompunerea ecranului intr-o astfel de fereastra si ferestre invecinate. In acest din urma caz , ferestrele invecinate ar suferi comprimari serioasa.

Dar , despre toate acestea , poate intr-un numar viitor.

LISTING 1

```
   0>REM ? 1987 VIDEO BYTE STUDIOS
   2 RESTORE 17
   5 PAPER 0: BORDER 0: INK 4: BRIGHT 0
  10 CLEAR 39999: PRINT AT 21,29;"VBS": LOAD ""CODE 47e3: LOAD ""CODE 6e4: RANDO
MIZE USR 48336: RANDOMIZE USR 48359
  12 BORDER 0
  15 DIM a(6): DIM b(6): DIM q$(6,22): DIM c(6)
  16 FOR a=1 TO 6: READ q$(a): NEXT a
  17 DATA "PARTICULARA NORMALA","PARTICULARA ORIZONTALA","PARTICULARA VERTICALA"
,"GENERALA NORMALA","GENERALA ORIZONTALA","GENERALA VERTICALA"
  20 POKE 23606,208: POKE 23607,184
  22 LET attr=23205: LET att1=BIN 00000010: LET att2=BIN 01000111: LET mode=1
  23 CLS
  25 PRINT AT 1,6; INK 6; PAPER 0; BRIGHT 0;"COMPRESOR DE ECRAN"
  27 PRINT INK 5; PAPER 0; BRIGHT 0;AT 5,1;"1 UNITATE DE SALVARE/INCARCARE";AT 7
,1;"2 COMPRIMARE NORMALA";AT 9,1;"3 COMPRIMARE ORIZONTALA";AT 11,1;"4 COMPRIMARE
 VERTICALA";AT 13,1;"5 TEST";AT 15,1;"6 COMPRIMARE";AT 17,1;"7 DECOMPRIMARE"
  29 GO SUB 9000
  30 PRINT AT 21,1; INK 2; PAPER 0; BRIGHT 0;"APASATI TASTA CORESPUNZATOARE"
  40 LET a$=INKEY$
  50 GO TO (a$="1")*1000+(a$="2")*2000+(a$="3")*3000+(a$="4")*4000+(a$="5")*5000
+(a$="6")*6000+(a$="7")*9900+(a$<>"1" AND a$<>"2" AND a$<>"3" AND a$<>"4" AND a$
<>"5" AND a$<>"6" AND a$<>"7" AND a$<>" " AND A$<>"8")*400+(a$=" ")*8000
 400 POKE attr,att1: POKE attr+1,att2
 410 LET attr=attr+1: IF attr=23230 THEN POKE 23230,att1: LET attr=23200
 420 GO TO 40
1000 CLS : PRINT AT 0,2; INK 2; BRIGHT 1;"UNITATE DE SALVARE/INCARCARE"; INK 5;
BRIGHT 0;AT 7,1;"1 SALVARE A PROGRAMULUI";AT 9,1;"2 SALVARE A DECOMPRESORULUI";A
T 11,1;"3 INCARCARE A UNUI NOU ECRAN";AT 21,2; INK 7; BRIGHT 1;"Apasati 'SPACE'
pentru meniu"
1020 LET a$=INKEY$: GO TO (a$<>"1" AND a$<>"2" AND a$<>"3" AND a$<>" ")*1020+(a$
="1")*1100+(a$="2")*1200+(a$="3")*1300+(a$=" ")*17
1100 CLEAR : CLS : PRINT AT 0,0; INK 2;"SALVARE A PROGRAMULUI COMPRESOR": SAVE C
HR$ 22+CHR$ 1+CHR$ 0+"COM.BAS" LINE 0: SAVE CHR$ 22+CHR$ 19+CHR$ 0+"COM.OBJ"CODE
 47000,1970: SAVE CHR$ 22+CHR$ 21+CHR$ 0+"DEC.OBJ"CODE 60000,550: PRINT AT 10,6;
 INK 6; BRIGHT 1;"SALVARE EXECUTATA !";AT 21,1; INK 7; BRIGHT 1;"Apasati 'SPACE'
 pentru meniu"
1110 IF INKEY$<>" " THEN GO TO 1110
1120 RUN 15
1200 CLS : PRINT AT 0,0; INK 2; BRIGHT 1;"SALVARE A RUTINEI DECOMPRESOARE";AT 5,
0; INK 5;" Rutina este relocabila si poatefi incarcata ulterior folosind  "; INK
 6;"LOAD """"CODF adresa de start"' INK 5;" Folositi "; INK 6'" POKE adresa de s
tart +424,low   POKE adresa de start +425,high"; INK 5'"pentru a transfera adres
a de la care incepe fisierul comprimat  si"; INK 6;" RANDOMIZE USR adresa de sta
rt"; INK 5;"pentru decomprimare."
```

```
1210 SAVE CHR$ 22+CHR$ 1+CHR$ 0+"DEC.OBJ"CODE 60000,550: PRINT AT 21,2; INK 7; B
RIGHT 1;"Apastai 'SPACE' pentru meniu"
1220 IF INKEY$<>" " THEN GO TO 1220
1230 GO TO 17
1300 CLS : REM
1500 PRINT AT 0,12; INVERSE 1;"INCARCARE";AT 10,0; INK 7;" Dupa incarcare apasat
i 'SPACE' ": RANDOMIZE USR 48377
1508 RANDOMIZE USR 47040
1509 PRINT AT 21,6; INK 2; PAPER 7;; INVERSE 1;"INCARCARE TERMINATA": FOR W=1 TO
 10: NEXT W: PRINT AT 21,6; INK 2; PAPER 7; INVERSE 0;"INCARCARE TERMINATA": FOR
 W=1 TO 10: NEXT W
1510 LET a$=INKEY$
1520 IF a$<>" " THEN GO TO 1509
1540 CLS : GO TO 25
2000 RANDOMIZE USR 47028
2010 IF mode=2 THEN GO TO 2500
2015 RANDOMIZE USR 47234
2020 CLS : PRINT AT 0,5; INK 2;"COMPRIMARE INCHEIATA"'' INK 5;"Lungimea fisierul
ui : ";PEEK 47284+256*PEEK 47285
2030 PRINT ''' INK 5;"SE SALVEAZA FISIERUL ?"
2040 LET a$=INKEY$
2050 IF a$<>"d" AND a$<>"D" AND a$<>"n" AND a$<>"N" THEN GO TO 2040
2060 IF a$="n" OR a$="N" THEN CLS : GO TO 23
2070 INPUT "Numele de fisier ? ";n$
2080 IF n$="" THEN LET n$="V.B.S. SCREEN$ "
2090 CLS : PRINT AT 10,0; INK 6;"Se salveaza fisierul "; INK 5;n$
2100 SAVE n$CODE 50000,PEEK 47284+256*PEEK 47285
2110 CLS : GO TO 23
2500 RANDOMIZE USR 47001: IF PEEK 47000=0 THEN CLS : PRINT AT 5,0; INK 5;"COD DE
 ESCAPE INEXISTENT !": GO TO 8010
2503 RANDOMIZE USR 47490
2505 CLS : PRINT AT 0,5; INK 2;"COMPRIMARE INCHEIATA"'': LET LENGTH=PEEK 47284+2
56*PEEK 47285: PRINT INK 5;"Lungimea fisierului : ";LENGTH
2510 PRINT ''' INK 5;"Se salveaza fisierul ?"
2520 LET A$=INKEY$
2530 IF A$="d" OR A$="D" THEN GO TO 2550
2540 IF A$="n" OR A$="N" THEN CLS : GO TO 23
2545 GO TO 2520
2550 INPUT "Numele de fisier ? ";N$
2560 IF N$="" THEN LET N$="V.B.S. SCREEN$ "
2570 CLS : PRINT AT 10,0; INK 6;"Se salveaza fisierul "; INK 5;n$
2600 SAVE n$CODE 50000,length
2610 CLS : GO TO 23
3000 RANDOMIZE USR 47028
3010 IF mode=2 THEN GO TO 3500
3020 RANDOMIZE USR 47243
3030 GO TO 2020
3500 RANDOMIZE USR 47001
3510 IF PEEK 47000=0 THEN CLS : PRINT INK 5;AT 5,0;"COD DE ESCAPE INEXISTENT !":
 GO TO 8010
3520 RANDOMIZE USR 47499: GO TO 2505
4000 RANDOMIZE USR 47028
4010 IF mode=2 THEN GO TO 4500
4020 RANDOMIZE USR 47252
4030 GO TO 2020
4500 RANDOMIZE USR 47001
4510 IF PEEK 47000=0 THEN CLS : PRINT AT 5,0; INK 5;"COD DE ESCAPE INEXISTENT !"
: GO TO 8010
4520 RANDOMIZE USR 47503: GO TO 2505
5000 FOR a=1 TO 6: LET c(a)=0: NEXT a: RANDOMIZE USR 47028
5010 RANDOMIZE USR 47001
5020 RANDOMIZE USR 47234
5030 LET a(1)=PEEK 47284+256*PEEK 47285
5040 RANDOMIZE USR 47243: LET a(2)=PEEK 47284+256*PEEK 47285
5050 RANDOMIZE USR 47252: LET a(3)=PEEK 47284+256*PEEK 47285
5060 IF PEEK 47000=0 THEN GO TO 5500
5070 RANDOMIZE USR 47490: LET a(4)=PEEK 47284+256*PEEK 47285
5080 RANDOMIZE USR 47499: LET a(5)=PEEK 47284+256*PEEK 47285
5090 RANDOMIZE USR 47503: LET a(6)=PEEK 47284+256*PEEK 47285
5100 FOR a=1 TO 6: LET b(a)=a(a): NEXT a
5110 FOR a=1 TO 5
5120 IF b(a)>b(a+1) THEN LET xx=b(a): LET b(a)=b(a+1): LET b(a+1)=xx: GO TO 5110
5130 NEXT a
5500 CLS : PRINT AT 0,8; INK 2;"TEST INCHEIAT"
5510 IF PEEK 47000<>0 THEN PRINT INK 5''"Codul de escape gasit : ";PEEK 47000''
5520 PRINT '' INK 5;" COMPRIMAREA        LUNGIMEA      ___________     ______
"''
5530 FOR a=1 TO 6
5540 FOR b=1 TO 6
5550 IF b(a)=a(b) AND c(b)=0 THEN PRINT INK 6;q$(b);" ";a(b): LET c(b)=1
5560 NEXT b
5570 NEXT a
5580 PRINT AT 21,2; INK 7; BRIGHT 1;"Apasati 'SPACE' pentru meniu"
5590 IF INKEY$<>" " THEN GO TO 5590
5600 CLS : GO TO 23
6000 IF mode=1 THEN LET mode=2: PAUSE 0: GO SUB 9000: GO TO 40
6010 LET mode=1: PAUSE 0: GO SUB 9000: GO TO 40
8000 RANDOMIZE USR 47028
8010 LET a$=INKEY$
8020 IF a$<>" " THEN GO TO 8010
8030 CLS : GO TO 23
9000 IF mode=1 THEN PRINT INK 5; PAPER 0; BRIGHT 0;AT 15,14;"PARTICULARA": RETUR
N
9010 PRINT INK 5; PAPER 0; BRIGHT 0;AT 15,14;"GENERALA   ": RETURN
9900 CLS : PRINT INK 4;AT 5,0;"Apasati 'SPACE' pentru a incepe"''"Dupa executie
apasati 'SPACE'      din nou pentru meniu"
9910 IF INKEY$<>" " THEN GO TO 9910
```

```
9920 RANDOMIZE USR 48420
9930 IF INKEY$<>" " THEN GO TO 9930
9940 GO TO 17
LISTING 2

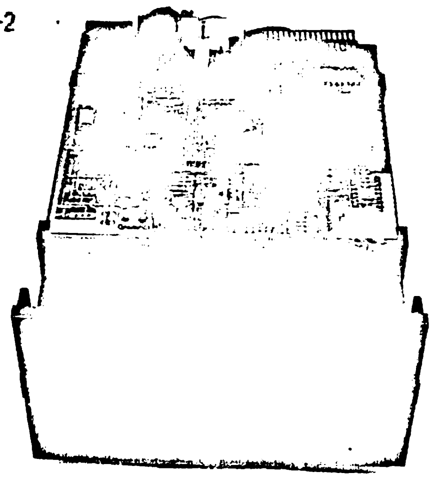


** pass         1
** errors       0
** warnings     0
                  *LLIST    ON
B798=          5        ORG  47000
639C=                   PUT  25500
                  ;
B798 00        10 ESCAPE.CODE.C:
                        DEFB 0
                  ;
                  ;ESCAPE > scans the display to find the esca
pe character. Once found ,this  is stored in the variable ESCAPE
.CODE.C
                  ;
B799 AF           ESCAPE:
                        XOR  A
B79A 3298B7             LD   (ESCAPE.CODE.C),A ; no   such cod
e
B79D 3E01               LD   A,1
B79F 01001B       scan.loop:
                        LD   BC,6912
B7A2 210040             LD   HL,16384
B7A5 EDB1               CPIR
B7A7 C2B0B7             JP   NZ,found
B7AA 3C                 INC  A
B7AB B7                 OR   A
B7AC C8                 RET  Z
B7AD C39FB7             JP   scan.loop
B7B0 3298B7       found:LD   (ESCAPE.CODE.C),A
B7B3 C9                 RET
               20 ;
                  ;MOVE.FROM.MEMORY.TO.SCREEN > service routin
e which moves the work file in the screen area
                  ;
B7B4 21409C       MOVE.FROM.MEMORY.TO.SCREEN:
                        LD   HL,40000
B7B7 110040             LD   DE,16384
B7BA 01001B             LD   BC,6912
B7BD EDB0               LDIR
B7BF C9                 RET
```

```
               30 ;
                  ;MOVE.FROM.SCREEN.TO.MEMORY > service routin
e which stores the current display file
                  ;
B7C0 210040       MOVE.FROM.SCREEN.TO.MEMORY:
                        LD   HL,16384
B7C3 11409C             LD   DE,40000
B7C6 01001B             LD   BC,6912
B7C9 EDB0               LDIR
B7CB C9                 RET
               40 ;
                  ;COMPRESS.PECULIAR > special loop used by th
e 3 types of peculiar comps.
                  ;
B7CC DD2151C3     COMPRESS.PECULIAR:
                        LD   IX,compressing.space+1 ; IX point
s to the compressed file
B7D0 210040             LD   HL,16384 ; HL holds the screen ad
dress
B7D3 110100             LD   DE,1 ; DE counts the length of th
e compressed file
B7D6 010000             LD   BC,0 ; BC holds the current line/
column values
B7D9 7E           label.1:
                        LD   A,(HL)
B7DA DD7700       label.2:
                        LD   (IX+0),A
B7DD DD23               INC  IX
B7DF 13                 INC  DE
B7E0 B7                 OR   A
B7E1 CAF2B7             JP   Z,label.3
B7E4 FEFF               CP   255
B7E6 CA2BB8             JP   Z,label.4
B7E9 CD0000       modify.0:
                        CALL 0
B7EC FEFF               CP   255
B7EE C8                 RET  Z
B7EF C3D9B7             JP   label.1
B7F2 3E01         label.3:
                        LD   A,1
B7F4 08                 EX   AF,AF'
B7F5 CD0000       label.5:
                        CALL 0
B7F8 FEFF               CP   255
B7FA CA6BB9             JP   Z,EXIT.1
B7FD 7E                 LD   A,(HL)
B7FE FE00               CP   0
B800 CA0EB8             JP   Z,label.6
B803 08                 EX   AF,AF'
```

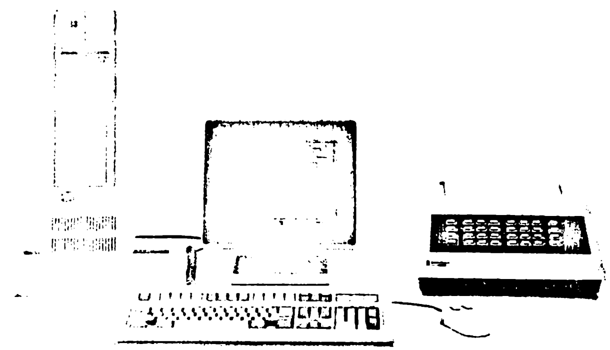
```
B804 DD7700                 LD   (IX+0),A
B807 DD23                   INC  IX
B809 13                     INC  DE
B80A 08                     EX   AF,AF'
B80B C3DAB7                 JP   label.2
B80E 08             label.6:
                            EX   AF,AF'
B80F 3C                     INC  A
B810 FEFF                   CP   255
B812 CA19B8                 JP   Z,label.7
B815 08                     EX   AF,AF'
B816 C3F5B7                 JP   label.5
B819 DD7700         label.7:
                            LD   (IX+0),A
B81C DD23                   INC  IX
B81E 13                     INC  DE
B81F CD0000         modify.1:
                            CALL 0
B822 FEFF                   CP   255
B824 C8                     RET  Z
B825 C3D9B7                 JP   label.1
B828 3E01           label.4:
                            LD   A,1
B82A 08                     EX   AF,AF'
B82B CD0000         label.8:
                            CALL 0
B82E FEFF                   CP   255
B830 CAB6B9                 JP   Z,EXIT.1
B833 7E                     LD   A,(HL)
B834 FEFF                   CP   255
B836 CA44B8                 JP   Z,label.9
B839 08                     EX   AF,AF'
B83A DD7700                 LD   (IX+0),A
B83D DD23                   INC  IX
B83F 13                     INC  DE
B840 08                     EX   AF,AF'
B841 C3DAB7                 JP   label.2
B844 08             label.9:
                            EX   AF,AF'
B845 3C                     INC  A
B846 FEFF                   CP   255
B848 CA19B8                 JP   Z,label.7
B84B 08                     EX   AF,AF'
B84C C32BB8                 JP   label.8
                    ;
                    ;COMPRESS.PECULIAR.NORMAL.PIXEL >   applies
 the peculiar normal compression  only to the pixel file
                    ;
B84F DD2150C3       COMPRESS.PECULIAR.NORMAL.PIXEL:
                            LD   IX,compressing.space
B853 2181BD                 LD   HL,NEXT.PIXEL.NORMAL
B856 AF                     XOR  A
B857 C36FB8                 JP   common.1
                    ;
                    ;COMPRESS.PECULIAR.HORIZONTAL.PIXEL > applie
s the peculiar horizontal comp.  only to the pixel file
                    ;
B85A DD2150C3       COMPRESS.PECULIAR.HORIZONTAL.PIXEL:
                            LD   IX,compressing.space
B85E 2189BD                 LD   HL,NEXT.PIXEL.HORIZONTAL
B861 3E01                   LD   A,1
B863 C36FB8                 JP   common.1
                    ;
                    ;COMPRESS.PECULIAR.VERTICAL.PIXEL > applies
the peculiar vertical comp. only to the pixel file
                    ;
B866 DD2150C3       COMPRESS.PECULIAR.VERTICAL.PIXEL:
                            LD   IX,compressing.space
B86A 219FBD                 LD   HL,NEXT.PIXEL.VERTICAL
B86D 3E02                   LD   A,2
                    ;
B86F DD7700         common.1:
                            LD   (IX+0),A
B872 22EAB7                 LD   (modify.0+1),HL
B875 2220B8                 LD   (modify.1+1),HL
B878 22F6B7                 LD   (label.5+1),HL
B87B 222CB8                 LD   (label.8+1),HL
B87E CDCCB7                 CALL COMPRESS.PECULIAR
B881 C9                     RET
                    ;
                    ;THE NEXT THREE ENTRY POINTS ARE    REFERING
 TO THE WHOLE DISPLAY FILE !
                    ;
B882 CD4FB8         COMPRESS.PECULIAR.NORMAL:
                            CALL COMPRESS.PECULIAR.NORMAL.PIXEL
B885 21B5BD                 LD   HL,NEXT.ATTRIBUTE.HORIZONTAL
B888 C39AB8                 JP   common.2
                    ;
B88B CD5AB8         COMPRESS.PECULIAR.HORIZONTAL:
                            CALL COMPRESS.PECULIAR.HORIZONTAL.PIXE
L
B88E 21B5BD                 LD   HL,NEXT.ATTRIBUTE.HORIZONTAL
B891 C39AB8                 JP   common.2
                    ;
B894 CD66B8         COMPRESS.PECULIAR.VERTICAL:
                            CALL COMPRESS.PECULIAR.VERTICAL.PIXEL
B897 21BDBD                 LD   HL,NEXT.ATTRIBUTE.VERTICAL
                    ;
```

```
B89A 010000         common.2:
                            LD   BC,0
B89D 22EAB7                 LD   (modify.0+1),HL
B8A0 2220B8                 LD   (modify.1+1),HL
B8A3 22F6B7                 LD   (label.5+1),HL
B8A6 222CB8                 LD   (label.8+1),HL
B8A9 210058                 LD   HL,22528
B8AC CD09B7                 CALL label.1
B8AF EB                     EX   DE,HL
B8B0 22B4B8                 LD   (LENGTH),HL
B8B3 C9                     RET
                    90 ;
C350=               compressing.space:
                            EQU  50000
B8B4 0000           LENGTH:
                            DEFW 0
                    ;
                    ;COMPRESS.GENERAL > special loop    used by
the general compressing routines
                    ;
B8B6 DD2152C3       COMPRESS.GENERAL:
                            LD   IX,compressing.space+2
B8BA 010000                 LD   BC,0
B8BD 210040                 LD   HL,16384
B8C0 110200                 LD   DE,2
B8C3 7E             label.10:
                            LD   A,(HL)
B8C4 32B5B9         label.11:
                            LD   (last.value),A
B8C7 DD7700                 LD   (IX+0),A
B8CA DD23                   INC  IX
B8CC 13                     INC  DE
B8CD CD0000         modify.2:
                            CALL 0
B8D0 FEFF                   CP   255
B8D2 C8                     RET  Z
B8D3 7E                     LD   A,(HL)
B8D4 D5                     PUSH DE
B8D5 5F                     LD   E,A
B8D6 3AB5B9                 LD   A,(last.value)
B8D9 BB                     CP   E
B8DA CAE2B8                 JP   Z,label.12
B8DD 7B                     LD   A,E
B8DE D1                     POP  DE
B8DF C3C4B8                 JP   label.11
B8E2 D1             label.12:
                            POP  DE
B8E3 CD0000         modify.3:
```



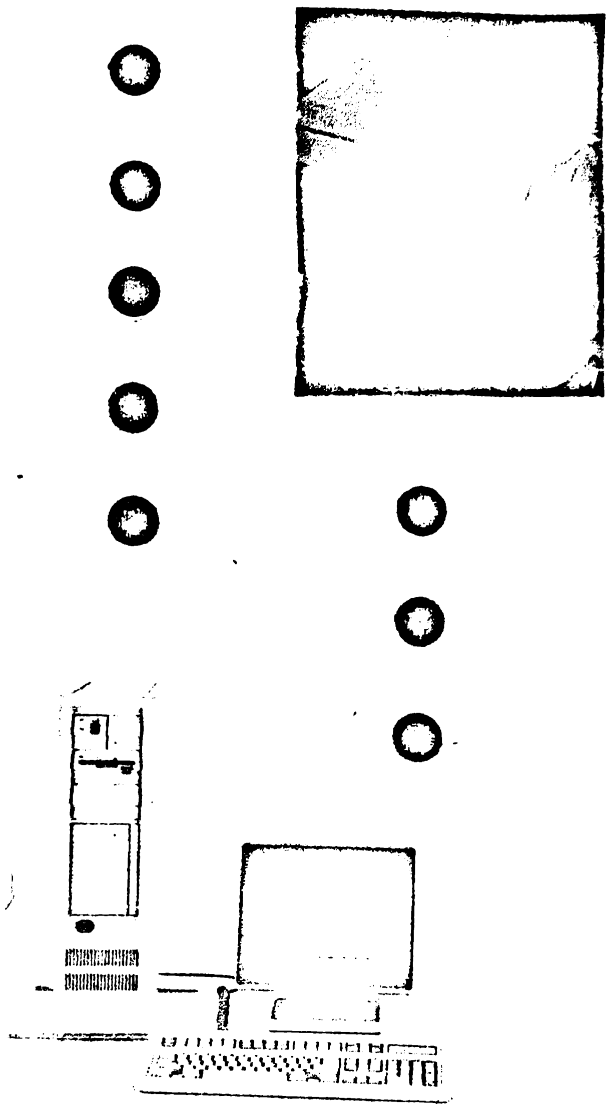
```
                            CALL 0
B8E6 FEFF                   CP   255
B8E8 CABEB9                 JP   Z,EXIT.2
B8EB D5                     PUSH DE
B8EC 3AB5B9                 LD   A,(last.value)
B8EF 5F                     LD   E,A
B8F0 7E                     LD   A,(HL)
B8F1 BB                     CP   E
B8F2 CA00B9                 JP   Z,label.13
B8F5 7B                     LD   A,E
B8F6 DD7700                 LD   (IX+0),A
B8F9 DD23                   INC  IX
B8FB D1                     POP  DE
B8FC 13                     INC  DE
B8FD C3C3B8                 JP   label.10
B900 D1             label.13:
                            POP  DE
B901 3E03                   LD   A,3
B903 08                     EX   AF,AF'
B904 3A93B7                 LD   A,(ESCAPE.CODE.C)
B907 DD7700                 LD   (IX+0),A
B90A DD23                   INC  IX
B90C 13                     INC  DE
B90D CD0000         label.14:
                            CALL 0
B910 FEFF                   CP   255
B912 CAC8B9                 JP   Z,EXIT.3
B915 D5                     PUSH DE
B916 7E                     LD   A,(HL)
B917 5F                     LD   E,A
B918 3AB5B9                 LD   A,(last.value)
B91B BB                     CP   E
B91C CA2BB9                 JP   Z,label.15
B91F D1                     POP  DE
B920 08                     EX   AF,AF'
B921 DD7700                 LD   (IX+0),A
B924 DD23                   INC  IX
B926 13                     INC  DE
B927 08                     EX   AF,AF'
B928 C3C3B8                 JP   label.10
B92B D1             label.15:
                            POP  DE
B92C 08                     EX   AF,AF'
B92D 3C                     INC  A
B92E FEFF                   CP   255
B930 CA37B9                 JP   Z,label.16
B933 08                     EX   AF,AF'
B934 C30DB9                 JP   label.14
```

```
B937 DD7700         label.16:
                            LD    (IX+0),A
B93A DD23                   INC   IX
B93C 13                     INC   DE
B93D 08                     EX    AF,AF'
B93E CD0000         modify.4:
                            CALL  0
B941 FEFF                   CP    255
B943 C8                     RET   Z
B944 7E                     LD    A,(HL)
B945 C3C4B8                 JP    label.11
                    ;
                    ;THE FOLLOWING 3 ENTRY POINTS ARE   REFERING
 ONLY TO THE PIXEL FILE !
                    ;
B948 DD2150C3       COMPRESS.GENERAL.NORMAL.PIXEL:
                            LD    IX,compressing.space
B94C 2181BD                 LD    HL,NEXT.PIXEL.NORMAL
B94F 3E03                   LD    A,3
B951 C369B9                 JP    common.3
                    ;
B954 DD2150C3       COMPRESS.GENERAL.HORIZONTAL.PIXEL:
                            LD    IX,compressing.space
B958 2189BD                 LD    HL,NEXT.PIXEL.HORIZONTAL
B95B 3E04                   LD    A,4
B95D C369B9                 JP    common.3
                    ;
B960 DD2150C3       COMPRESS.GENERAL.VERTICAL.PIXEL:
                            LD    IX,compressing.space
B964 219FBD                 LD    HL,NEXT.PIXEL.VERTICAL
B967 3E05                   LD    A,5
                    ;
B969 DD7700         common.3:
                            LD    (IX+0),A
B96C 3A98B7                 LD    A,(ESCAPE.CODE.C)
B96F DD7701                 LD    (IX+1),A
B972 22CEB8                 LD    (modify.2+1),HL
B975 22E4B8                 LD    (modify.3+1),HL
B978 223FB9                 LD    (modify.4+1),HL
B97B 220EB9                 LD    (label.14+1),HL
B97E CDB6B8                 CALL  COMPRESS.GENERAL
B981 C9                     RET
                    ;
                    ;THE FOLLOWING 3 ENTRY POINTS ARE   REFERING
 AT THE WHOLE DISPLAY FILE !
                    ;
B982 CD48B9         COMPRESS.GENERAL.NORMAL:
                            CALL  COMPRESS.GENERAL.NORMAL.PIXEL
B985 21B5BD                 LD    HL,NEXT.ATTRIBUTE.HORIZONTAL
B988 C39AB9                 JP    common.4
                    ;
B98B CD54B9         COMPRESS.GENERAL.HORIZONTAL:
                            CALL  COMPRESS.GENERAL.HORIZONTAL.PIXEL
B98E 21B5BD                 LD    HL,NEXT.ATTRIBUTE.HORIZONTAL
B991 C39AB9                 JP    common.4
                    ;
B994 CD60B9         COMPRESS.GENERAL.VERTICAL:
                            CALL  COMPRESS.GENERAL.VERTICAL.PIXEL
B997 21BDBD                 LD    HL,NEXT.ATTRIBUTE.VERTICAL
                    ;
B99A 010000         common.4:
                            LD    BC,0
B99D 22CEB8                 LD    (modify.2+1),HL
```



```
B9A0 22E4B8                 LD    (modify.3+1),HL
B9A3 223FB9                 LD    (modify.4+1),HL
B9A6 220EB9                 LD    (label.14+1),HL
B9A9 210058                 LD    HL,22528
B9AC 7E                     LD    A,(HL)
B9AD CDC4B8                 CALL  label.11
B9B0 EB                     EX    DE,HL
B9B1 22B4B3                 LD    (LENGTH),HL
B9B4 C9                     RET
                    ;
B9B5 00             last.value:
                            DEFB  0
                    ;
```

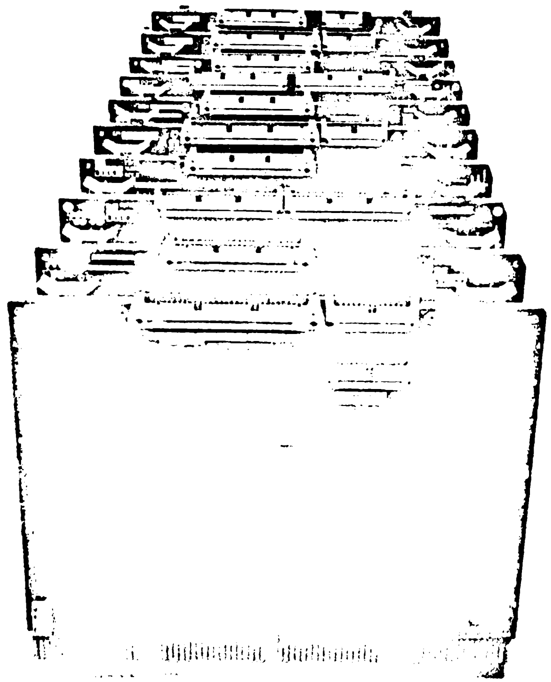
```
B9B6 08            EXIT.1:
                           EX    AF,AF'
B9B7 DD7700                LD    (IX+0),A
B9BA DD23                  INC   IX
B9BC 13                    INC   DE
B9BD C9                    RET
                   ;
B9BE 3AB5B9        EXIT.2:
                           LD    A,(last.value)
B9C1 DD7700                LD    (IX+0),A
B9C4 DD23                  INC   IX
B9C6 13                    INC   DE
B9C7 C9                    RET
                   ;
B9C8 08            EXIT.3:
                           EX    AF,AF'
B9C9 DD7700                LD    (IX+0),A
B9CC DD23                  INC   IX
B9CE 13                    INC   DE
B9CF C9                    RET
                   ;
B9D0               FONT :DEFS 768
                   ;
BCD0 21003D        FONT.CONSTRUCTOR:
                           LD    HL,15616
BCD3 11D0B9                LD    DE,FONT
BCD6 010003                LD    BC,768
BCD9 7E            font.loop:
                           LD    A,(HL)
BCDA 1F                    RRA
BCDB B6                    OR    (HL)
BCDC CBBF                  RES   7,A
BCDE 12                    LD    (DE),A
BCDF 23                    INC   HL
BCE0 13                    INC   DE
BCE1 0B                    DEC   BC
BCE2 78                    LD    A,B
BCE3 B1                    OR    C
BCE4 20F3                  JR    NZ,font.loop
BCE6 C9                    RET
                   ;
BCE7 2160EA        ARTIFICIAL.RELOCATION:
                           LD    HL,60000
BCEA 1124BD                LD    DE,DECOMPRESSOR.SPACE
BCED 012602                LD    BC,550
BCF0 EDB0                  LDIR
BCF2 0124BD                LD    BC,DECOMPRESSOR.SPACE
BCF5 C324BD                JP    DECOMPRESSOR.SPACE
```

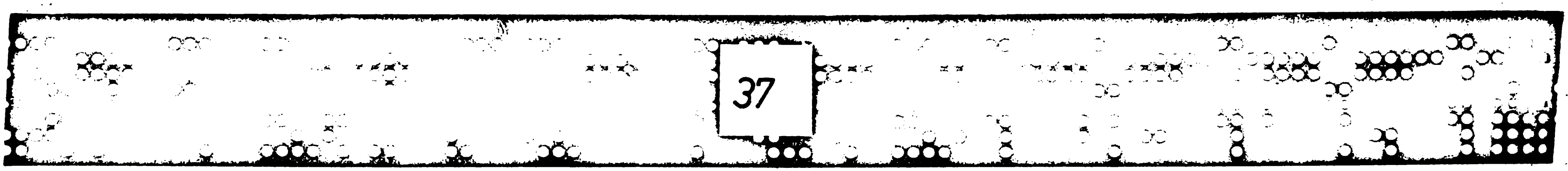


```
BCF8 C9                    RET
                   ;
BCF9 F3            LOAD.SCREEN:
                           DI
BCFA DD21FF3F              LD    IX,16383
BCFE 11011B                LD    DE,6913
BD01 37                    SCF
BD02 CD11BD                CALL  LOAD.BYTES.MODIFYIED
BD05 FB                    EI
BD06 7A                    LD    A,D
BD07 FE1A                  CP    #1A
BD09 C0                    RET   NZ
BD0A 7B                    LD    A,E
BD0B FEEE                  CP    #EE
BD0D C0                    RET   NZ
BD0E C3F9BC                JP    LOAD.SCREEN
                   ;
BD11 AF            LOAD.BYTES.MODIFYIED:
                           XOR   A
BD12 37                    SCF
BD13 08                    EX    AF,AF'
BD14 3E0F                  LD    A,15
BD16 D3FE                  OUT   (#FE),A
BD18 DBFE                  IN    A,(#FE)
BD1A 1F                    RRA
BD1B E620                  AND   32
BD1D F604                  OR    4
BD1F 4F                    LD    C,A
BD20 BF                    CP    A
BD21 C36B05                JP    #56B
BD24               DECOMPRESSOR.SPACE:
                           DEFS 550
                   ;
BDB5=              NEXT.ATTRIBUTE.HORIZONTAL:
                           EQU   48565
BDBD=              NEXT.ATTRIBUTE.VERTICAL:
                           EQU   48573
BD81=              NEXT.PIXEL.NORMAL:
                           EQU   48513
BD89=              NEXT.PIXEL.HORIZONTAL:
                           EQU   48521
BD9F=              NEXT.PIXEL.VERTICAL:
                           EQU   48543
                   ;
                   ;
BF4A 00            END.OF.FILE:
                           NOP
                   ;
```

```
** pass         2
** errors       0
** warnings     0
LISTING 3
```

```
** pass         1
** errors       0
** warnings     0
                   *LLIST    ON
0000=                    ORG  0
639C=                    PUT  25500
                   ;
                   ;
0000 21CD01        RELOCATOR:
                         LD   HL,RELOCATION.TABLE
0003 09                  ADD  HL,BC
0004 5E            loops:LD   E,(HL)
0005 23                  INC  HL
0006 56                  LD   D,(HL)
0007 23                  INC  HL
0008 7A                  LD   A,D
0009 B3                  OR   E
000A 2811                JR   Z,no.more.relocations
000C EB                  EX   DE,HL
000D 09                  ADD  HL,BC
000E D5                  PUSH DE
000F E5                  PUSH HL
0010 5E                  LD   E,(HL)
0011 23                  INC  HL
0012 56                  LD   D,(HL)
0013 EB                  EX   DE,HL
0014 09                  ADD  HL,BC
0015 EB                  EX   DE,HL
0016 E1                  POP  HL
0017 73                  LD   (HL),E
0018 23                  INC  HL
0019 72                  LD   (HL),D
001A E1                  POP  HL
001B 18E7                JR   loops
001D               no.more.relocations:
                   !40  :LD   HL,RELOCATOR
001D 210000        !40  :LD   HL,RELOCATOR
0020 110100        !41  :LD   DE,RELOCATOR+1
0023 012900              LD   BC,41
0026 3600                LD   (HL),0
0028 EDB0                LDIR
                   ;
002A C3A601        10   :JP   SELECTER
                   ;
                   ;
002D 78            ATTRIBUTE.LOCATOR:
                         LD   A,B
002E CB2F                SRA  A
0030 CB2F                SRA  A
0032 CB2F                SRA  A
0034 C658                ADD  A,#58
0036 67                  LD   H,A
0037 78                  LD   A,B
0038 E607                AND  7
003A 0F                  RRCA
003B 0F                  RRCA
003C 0F                  RRCA
003D 81                  ADD  A,C
003E 6F                  LD   L,A
003F C9                  RET
                   ;
0040 C5            PLOT.LOCATOR:
                         PUSH BC
0041 78                  LD   A,B
0042 E638                AND  56
0044 CB27                SLA  A
0046 CB27                SLA  A
0048 B1                  OR   C
0049 6F                  LD   L,A
004A 78                  LD   A,B
004B E607                AND  7
004D 67                  LD   H,A
004E 78                  LD   A,B
004F E6C0                AND  192
0051 CB3F                SRL  A
0053 CB3F                SRL  A
0055 CB3F                SRL  A
0057 84                  ADD  A,H
0058 C640                ADD  A,64
005A 67                  LD   H,A
005B C1                  POP  BC
005C C9                  RET
                   ;
005D 23            NEXT.PIXEL.NORMAL:
                         INC  HL
005E 7C                  LD   A,H
005F FE58                CP   #58
```

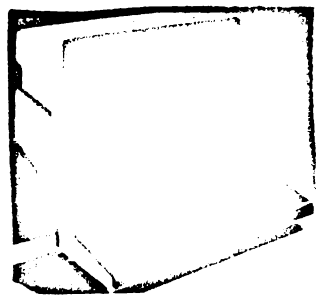




```
0061 C0              RET  NZ
0062 3EFF            LD   A,255
0064 C9              RET
                    ;
0065 0C             NEXT.PIXEL.HORIZONTAL:
                     INC  C
0066 79              LD   A,C
0067 FE20      xx1  :CP   32
0069 200B            JR   NZ,12
006B 0E00            LD   C,0
006D 04              INC  B
006E 78              LD   A,B
006F FEC0      xx2  :CP   192
0071 2003            JR   NZ,12
0073 3EFF            LD   A,255
0075 C9              RET
0076 CD4000    12   :CALL PLOT.LOCATOR
0079 AF              XOR  A
007A C9              RET
                    ;
007B 04             NEXT.PIXEL.VERTICAL:
                     INC  B
007C 78              LD   A,B
007D FEC0      xx3  :CP   192
007F 200B            JR   NZ,13
0081 0600            LD   B,0
0083 0C              INC  C
0084 79              LD   A,C
0085 FE20      xx4  :CP   32
0087 2003            JR   NZ,13
0089 3EFF            LD   A,255
008B C9              RET
008C CD4000    13   :CALL PLOT.LOCATOR
008F AF              XOR  A
0090 C9              RET
                    ;
0091 23             NEXT.ATTRIBUTE.HORIZONTAL:
                     INC  HL
0092 7C              LD   A,H
0093 FE5B            CP   91
0095 C0              RET  NZ
0096 3EFF            LD   A,255
0098 C9              RET
                    ;
0099 04             NEXT.ATTRIBUTE.VERTICAL:
                     INC  B
009A 78              LD   A,B
```

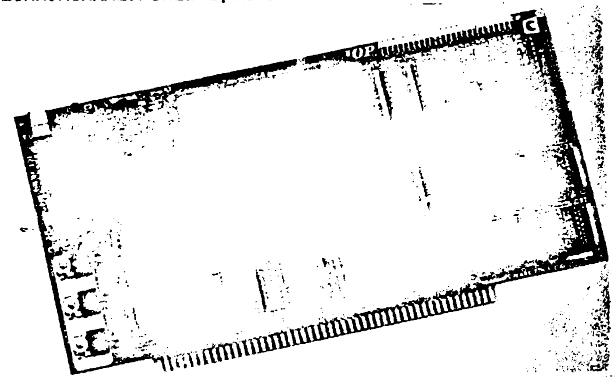
```
009B FE18            CP   24
009D 200B            JR   NZ,15
009F 0600            LD   B,0
00A1 0C              INC  C
00A2 79              LD   A,C
00A3 FE20            CP   32
00A5 2003            JR   NZ,15
00A7 3EFF            LD   A,255
00A9 C9              RET
00AA CD2D00    15   :CALL ATTRIBUTE.LOCATOR
00AD AF              XOR  A
00AE C9              RET
               150 ;
00AF 210040         DECOMPRESS.PECULIAR:
                     LD   HL,16384
00B2 010000          LD   BC,0
00B5 DD7E00         label.1:
                     LD   A,(IX+0)
00B8 77              LD   (HL),A
00B9 B7              OR   A
00BA 280E            JR   Z,label.2
00BC FEFF            CP   255
00BE 280A            JR   Z,label.2
00C0 DD23            INC  IX
00C2 CD0000         label.7:
                     CALL 0
00C5 FEFF            CP   255
00C7 C8              RET  Z
00C8 18EB            JR   label.1
00CA 5F             label.2:
                     LD   E,A
00CB DD23            INC  IX
00CD DD7E00          LD   A,(IX+0)
00D0 3D              DEC  A
00D1 280C            JR   Z,label.8
00D3 57              LD   D,A
00D4 CD0000         label.3:
                     CALL 0
00D7 FEFF            CP   255
00D9 C8              RET  Z
00DA 7B              LD   A,E
00DB 77              LD   (HL),A
00DC 15              DEC  D
00DD 20F5            JR   NZ,label.3
00DF DD23           label.8:
                     INC  IX
00E1 CD0000         label.6:
                     CALL 0
```

```
00E4 FEFF                CP   255
00E6 C8                  RET  Z
00E7 13CC                JR   label.1
          ;
00E9 210040    DECOMPRESS.GENERAL:
                         LD   HL,16384
00EC 010000              LD   BC,0
00EF 1806                JR   start.decompression
00F1 CD0000    label.4:
                         CALL 0
00F4 FEFF                CP   255
00F6 C8                  RET  Z
00F7 DD7E00    start.decompression:
                         LD   A,(IX+0)
00FA 5F                  LD   E,A
00FB 77                  LD   (HL),A
00FC DD23                INC  IX
00FE 3ACC01    t1   :LD   A,(ESCAPE.CODE.D)
0101 57                  LD   D,A
0102 DD7E00              LD   A,(IX+0)
0105 BA                  CP   D
0106 20E9                JR   NZ,label.4
0108 DD23                INC  IX
010A DD7E00              LD   A,(IX+0)
010D 57                  LD   D,A
010E 15                  DEC  D
010F CD0000    label.5:
                         CALL 0
0112 FEFF                CP   255
0114 C8                  RET  Z
0115 7B                  LD   A,E
0116 77                  LD   (HL),A
0117 15                  DEC  D
0118 20F5                JR   NZ,label.5
011A DD23                INC  IX
011C 1803                JR   label.4
          ;
011E 215D00    DECOMPRESS.PECULIAR.NORMAL.PIXEL:
                         LD   HL,NEXT.PIXEL.NORMAL
0121 1808                JR   common.1
               ;
0123 216500    DECOMPRESS.PECULIAR.HORIZONTAL.PIXEL:
                         LD   HL,NEXT.PIXEL.HORIZONTAl
0126 1803                JR   common.1
               ;
0128 217B00    DECOMPRESS.PECULIAR.VERTICAL.PIXEL:
                         LD   HL,NEXT.PIXEL.VERTICAl
012B 22C300    common.1:
                         LD   (label.7+1),HL
```



```
012E 22D500    t2   :LD   (label.3+1),HL
0131 22E200    t3   :LD   (label.6+1),HL
0134 C3AF00    t30  :JP   DECOMPRESS.PECULIAR
0137 DD23      DECOMPRESS.PECULIAR.NORMAL:
                     INC  IX
0139 CD1E01    t4   :CALL DECOMPRESS.PECULIAR.NORMAL.PIXEL
013C 219100    t5   :LD   HL,NEXT.ATTRIBUTE.HORIZONTAL
013F 1812            JR   common.2
0141 DD23      DECOMPRESS.PECULIAR.HORIZONTAL:
                     INC  IX
0143 CD2301    t6   :CALL DECOMPRESS.PECULIAR.HORIZONTAL.PI
XEL
0146 219100    t7   :LD   HL,NEXT.ATTRIBUTE.HORIZONTAL
0149 1808            JR   common.2
               ;
014B DD23      DECOMPRESS.PECULIAR.VERTICAL:
                     INC  IX
014D CD2801    t8   :CALL DECOMPRESS.PECULIAR.VERTICAL.PIXE
L
0150 219900    t9   :LD   HL,NEXT.ATTRIBUTE.VERTICAL
               ;
0153 22C300    common.2:
                     LD   (label.7+1),HL
0156 22D500    t10  :LD   (label.3+1),HL
0159 22E200    t11  :LD   (label.6+1),HL
015C 010000          LD   BC,0
015F 210058          LD   HL,22528
0162 C3B500    t12  :JP   label.1
               ;
0165 215D00    DECOMPRESS.GENERAL.NORMAL.PIXEL:
                     LD   HL,NEXT.PIXEL.NORMAL
0168 1808            JR   common.3
               ;
016A 216500    DECOMPRESS.GENERAL.HORIZONTAL.PIXEL:
                     LD   HL,NEXT.PIXEL.HORIZONTAL
016D 1803            JR   common.3
016F 217B00    DECOMPRESS.GENERAL.VERTICAL.PIXEL:
                     LD   HL,NEXT.PIXEL.VERTICAL
               ;
0172 22F200    common.3:
                     LD   (label.4+1),HL
0175 221001    t14  :LD   (label.5+1),HL
0178 C3E900    t15  :JP   DECOMPRESS.GENERAL
               ;
017B DD23      DECOMPRESS.GENERAL.NORMAL:
                     INC  IX
017D CD6501    t16  :CALL DECOMPRESS.GENERAL.NORMAL.PIXEL
0180 219100    t17  :LD   HL,NEXT.ATTRIBUTE.HORIZONTAL
0183 1812            JR   common.4
```

```
                         ;
0185 DD23                DECOMPRESS.GENERAL.HORIZONTAL:
                                INC  IX
0187 CD6A01              t18   :CALL DECOMPRESS.GENERAL.HORIZONTAL.PI
EL
018A 219100              t19   :LD   HL,NEXT.ATTRIBUTE.HORIZONTAL
018D 1808                       JR   common.4
                         ;
018F DD23                DECOMPRESS.GENERAL.VERTICAL:
                                INC  IX
0191 CD6F01              t20   :CALL DECOMPRESS.GENERAL.VERTICAL.PIXEL
0194 219900              t21   :LD   HL,NEXT.ATTRIBUTE.VERTICAL
                         ;
0197 22F200              common.4:
                                LD   (label.4+1),HL
019A 221001              t22   :LD   (label.5+1),HL
019D 010000                     LD   BC,0
01A0 210058                     LD   HL,22528
01A3 C3F700              t23   :JP   start.decompression
                         ;
01A6 DD2150C3            SELECTER:
                                LD   IX,50000
01AA DD7E00                     LD   A,(IX+0)
01AD B7                         OR   A
01AE 2887                       JR   Z,DECOMPRESS.PECULIAR.NORMAL
01B0 FE01                       CP   1
01B2 288D                       JR   Z,DECOMPRESS.PECULIAR.HORIZONTAL
01B4 FE02                       CP   2
01B6 2893                       JR   Z,DECOMPRESS.PECULIAR.VERTICAL
01B8 DD23                       INC  IX
01BA 08                         EX   AF,AF'
01BB DD7E00                     LD   A,(IX+0)
01BE 32CC01              t24   :LD   (ESCAPE.CODE.D),A
01C1 08                         EX   AF,AF'
01C2 FE03                       CP   3
01C4 28B5                       JR   Z,DECOMPRESS.GENERAL.NORMAL
01C6 FE04                       CP   4
01C8 28BB                       JR   Z,DECOMPRESS.GENERAL.HORIZONTAL
01CA 18C3                       JR   DECOMPRESS.GENERAL.VERTICAL
                         ;
01CC 00                  ESCAPE.CODE.D:
                                DEFB 0
                         ;
01CD 2B007700
     3D00AB00
     FF001F01
     24012901
     2C012F01
     32013A01
     3D014401
     47014E01
     51015401
     57015A01
     63016601
     6B017001
     73017601
     79017E01
     81018801
     8B019201
     95019801
     9B01A401
     35011E00
     2100BF01
     0000                RELOCATION.TABLE:
                                DEFW t0+1,t2+1,t3+1,t5+1,t1+1,DECOMPRE
SS.PECULIAR.NORMAL.PIXEL+1,DECOMPRESS.PECULIAR.HORIZONTAL.PIXEL+
```



```
1,DECOMPRESS.PECULIAR.VERTICAL.PIXEL+1,common.1+1,t2+1,t3+1,t4+1
,t5+1,t6+1,t7+1,t8+1,t9+1,common.2+1,t10+1,t11+1,t12+1,DECOMPRES
S.GENERAL.NORMAL.PIXEL+1,DECOMPRESS.GENERAL.HORIZONTAL.PIXEL+1,D
ECOMPRESS.GENERAL.VERTICAL.PIXEL+1,common.3+1,t14+1,t15+1,t16+1,
t17+1,t18+1,t19+1,t20+1,t21+1,common.4+1,t22+1,t23+1,t30+1,t40+1
,t41+1,t24+1,0
                         ;
021F 00                  END.OF.FILE:
                                NOP
                         ;

** pass        2
** errors      0
** warnings    0
```

AS.ING. MARIUS CRISAN

INREGISTRAREA SI REPRODUCEREA SEMNALELOR ANALOGICE CU MICROCALCULATORUL TIM-S

1. Introducere

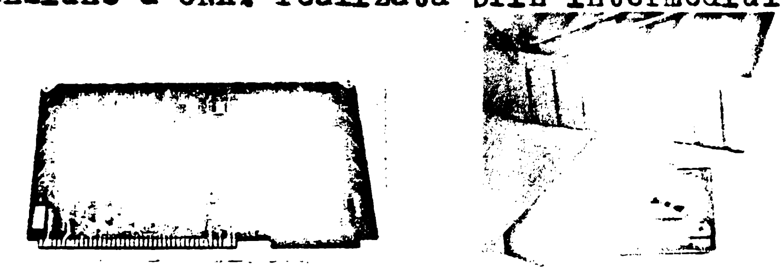
Conducerea proceselor tehnologice, de complexitate crescîndă la un nivel tot mai ridicat de eficiență, reclamă utilizarea din ce în ce mai largă a calculatoarelor. Legătura informațională, în sens cantitativ, dintre calculator și mărimile exterioare analogice ale procesului implică existența unei interfețe, al cărei element principal îl reprezintă convertorul analog-numeric (CAN). Datorită disponibilității, la un preț de cost scăzut, a convertoarelor numeric-analogice (CNA) integrate, au apărut ca eficiente o serie de soluții de utilizare a acestora pentru conversia analog-numerică. Dintre aceste soluții, în care CNA intervine într-o configurație cu reacție, avantajoasă într-o serie de aplicații, în care este necesar controlul unor mărimi analogice variabile în timp, este conversia analog-numerică cu urmărire /1/. Pe de altă parte, performanțele ridicate oferite de microcalculatoarele personale, raportate la prețul lor scăzut, au determinat lărgirea gamei de aplicație a acestora și în cadrul proceselor industriale.

In sensul celor menționate, prezenta lu-

crare se referă la o modalitate de interfaţare a unui CAN cu urmărire la un microcalculator TIM-S, soluţie ce oferă posibilităţi avantajoase de înregistrare şi reproducere, respectiv de prelucrare a semnalelor analogice.

2. Structura CAN cu urmărire şi interfaţarea cu microcalculatorul TIM-S

In fig.1 se prezintă schema de principiu a soluţiei de realizare a CAN şi modalitatea de conectare la microcalculator. Conversia analog-numerică se realizează pe principiul urmăririi, avînd la bază un CNA de 12 biţi, de tipul K 594 PA 1 (similar cu AD 562), un numărător reversibil de 12 biţi, realizat cu trei numărătoare sincrone, de tip CDB 4193 E şi un comparator de tensiune, de tip βM33o2. La intrările comparatorului se aplică ieşirea în tensiune a CNA, realizată prin intermediul



unui amplificator operaţional, de tip βM 3o1, precum şi mărimea de intrare analogică, care urmează a fi convertită /2/. Dacă mărimea analogică este mai mare decît ieşirea CNA, comparatorul va genera la ieşire un nivel logic coborît, determinînd incrementarea numărătorului cu o unitate. Procesul se repetă pînă în situaţia în care ieşirea CNA devine mai mare decît mărimea analogică de convertit, comparatorul generînd un nivel logic ridicat, ceea ce determină decrementarea cu o unitate a numărătorului. Comparatorul continuă să examineze diferenţa tensiunilor de la intrare, comandînd numărătorul într-un sens sau altul, în funcţie de rezultatul comparării. La echilibru, bucla de reacţie devine calată pe urmărirea variaţiei mărimii analogice de la intrare, atît timp cît nu se depăşeşte viteza de reacţie a buclei. In cazul unei mărimi de intrare continue, numărătorul, ale cărui ieşiri reprezintă aproximarea numerică a mărimii de intrare analogice, va alterna cu valoarea bitului cel mai puţin semnificativ. In fig.2 sînt reprezentate cronogramele specifice ciclului de conversie pentru acest caz.

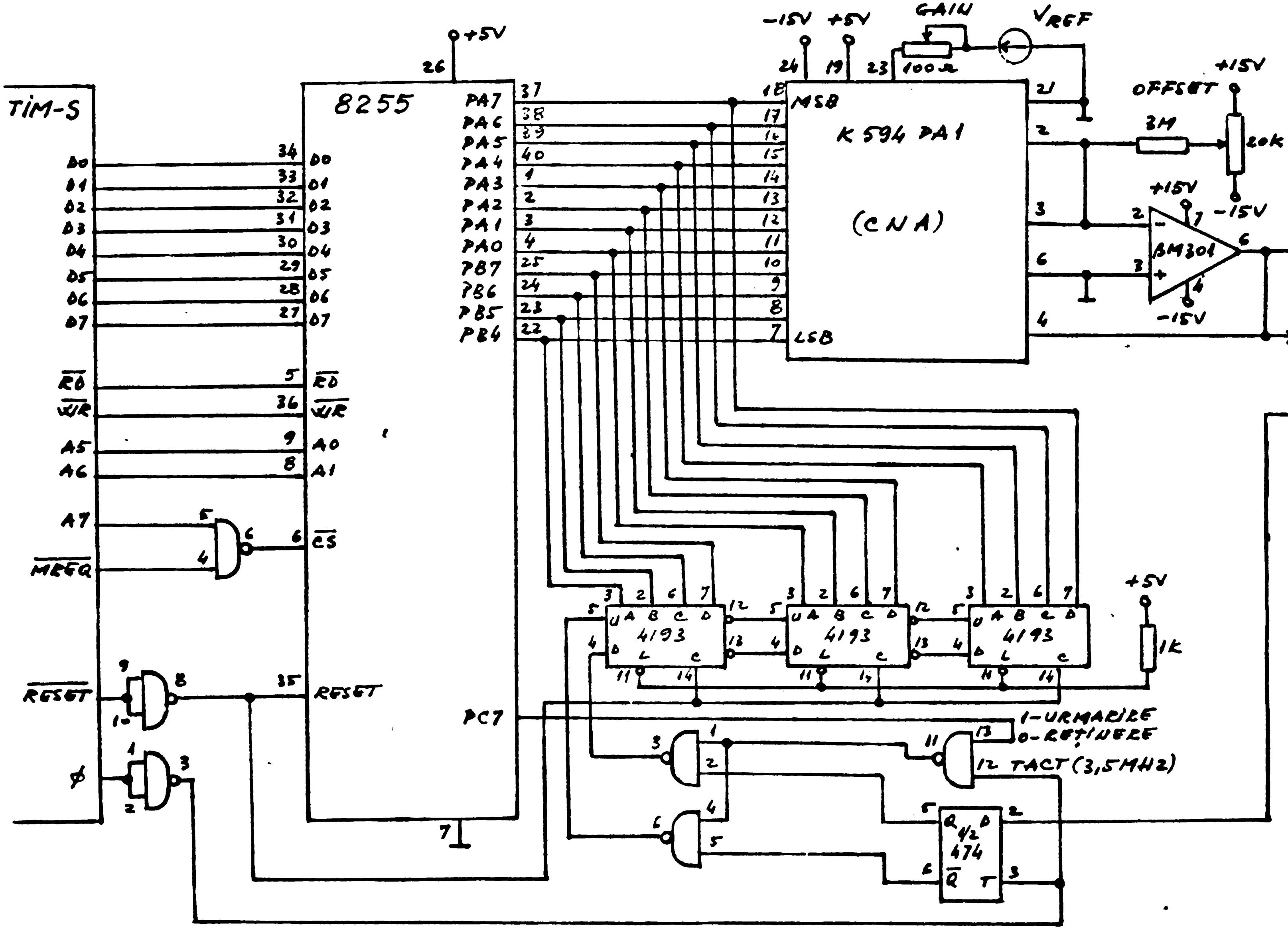


Fig.1. Schema de principiu a sistemului de conversie şi a interfeţei cu microcalculatorul

In bucla de reacţie s-a introdus un circuit basculant bistabil de tip D (1/2 CDB 474 E), între comparator şi intrarea numărătorului, pentru a asigura un timp adecvat de stabilire, între răspunsul comparatorului şi modificarea următoarei stări a numărătorului, dictat de timpul de răspuns al CNA şi al comparatorului.

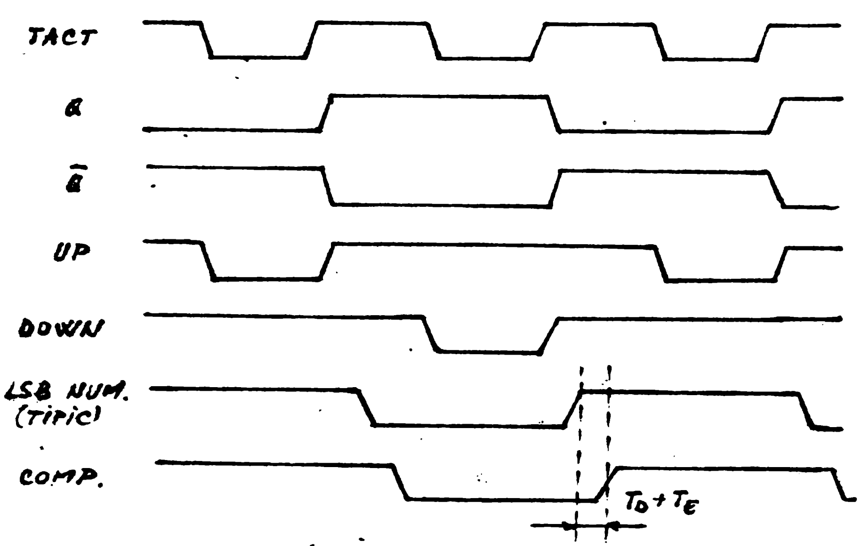


Fig.2. Cronogramele sistemului de conversie

Semnalul de tact al convertorului cu urmărire este asigurat de către calculatorul TIM-S, fiind tactul sistemului de 3,5 MH_z. In general, frecvenţa maximă de tact cu care se poate comanda convertorul este dată de relaţia:

$$ft_{max} = 1/(T_A + T_B + T_C + T_D + T_E) \ , \quad (1)$$

în care T_A reprezintă timpul de propagare al bistabilului, T_B este timpul minim de stabilire pentru numărător, T_C semnifică întîrzierea de propagare datorată numărătorului, T_D este timpul de răspuns al CNA (pentru n biţi), iar T_E timpul de răspuns al comparatorului.

Viteza maximă de reacţie a buclei este

$$LSR = ft \cdot V_{LSB} \ , \qquad (2)$$

unde V_{LSB} reprezintă tensiunea minimă, corespunzătoare bitului cel mai puţin semnificativ. In cazul în care $V_{REF} = 5V$, la o rezoluţie a convertorului de 12 biţi, rezultă pentru aplicaţia prezentată LSR=4,3 mv/µS.

In ceea ce priveşte erorile statice în c.c.. acestea pot fi atribuite numai com-

ponentelor analogice din schemă. Astfel, prezența tensiunii de offset la intrările amplificatorului operațional și comparatorului precum și variația ei cu temperatura (drift), fiind de ordinul de mărime a lui V_{LSB}, va afecta performanțele conversiei la capetele de scală. Ca urmare, se impune calibrarea atentă a decalajului de nul (offset) și a cîștigului (gain), cu ajutorul rezistențelor semireglabile aferente, (fig.1). Inainte de efectuarea calibrării, CNA trebuie adus la echilibru termic, fiind menținut alimentat cel puțin 30 min.

Monotonia este asigurată prin faptul că eroarea de neliniaritate a CNA este garantată la ± 1/2 LSB.

Interfațarea cu calculatorul TIM-S este realizată printr-un circuit de interfață programabil de tip 8255, prin intermediul căruia se poate preleva mărimea numerică pe 12 biți, ca rezultat al conversiei. De asemenea, în funcție de starea liniei PC7 se comandă convertorul pentru modul "urmărire" (PC7 = 1), respectiv "reținere" (PC7 = 0), în cadrul programului de achiziție.

In cazul în care se dorește achiziția cu o rezoluție de 8 biți, se va utiliza numai portul A al circuitului 8255.

Performanțele CAN astfel realizat sînt destul de ridicate, dacă se ține cont de prețul de cost scăzut al acestuia și facilitatea de implementare. Utilizînd frecvența de tact a calculatorului, au putut fi urmărite fără distorsiuni frecvențe de ordinul a 1 kH_z, cu amplitudinea vîrf la vîrf de 5 V. Evident, prin reducerea amplitudinii vîrf la vîrf, conversia prin urmărire poate continua pînă la valori mai mari ale frecvenței.

3. Aplicații de înregistrare și reproducere a semnalelor analogice

Beneficiind de facilitățile grafice și de memorare oferite de calculatorul TIM-S, se pot dezvolta o serie de aplicații, în care este necesară achiziția semnalelor de joasă frecvență.

Avantajul principal al CAN cu urmărire îl reprezintă faptul că nu necesită un circuit de eșantionare și reținere, rezultatul conversiei fiind disponibil în

mod continuu la ieşirile numărătorului şi, ca urmare, putînd fi memorat prin program, la orice moment de timp dorit, în limita vitezei de prelevare a ansamblului calculator-circuit de interfaţă programabil.

Considerînd,spre exemplu,numai achiziţia primilor 8 biţi mai semnificativi, un subprogram de memorare,a unui număr dat de eşantioane la anumite intervale de timp, poate fi elaborat simplu,într-o manieră ca cea din fig.3. Circuitul 8255 s-a programat în modul 0,cu porturile A şi B de intrare şi portul C de ieşire. Registrul dublu HL se încarcă cu adresa de început,de la care se doreşte memorarea unui număr de octeţi specificaţi prin registrul dublu DE.Intervalele de eşantionare sînt controlate prin decrementarea registrului B,iniţializat printr-o constantă de valoare corespunzătoare.

Dacă se doreşte reproducerea semnalului memorat,acesta va fi disponibil la ieşirea amplificatorului operaţional,prin utilizarea unei subrutine de genul celei anterior prezentate,(fig.4).

```
        DI
        LD      A, 92H
        OUT     (FF), A
        LD      HL, ADR
        LD      DE, NROCT
L2 :    LD      B, INT
        IN      A, (9F)
        LD      (HL), A
        INC     HL
L1 :    DJNZ    L1
        DEC     DE
        LD      A, D
        OR      E
        JR      NZ, L2
        EI
        RET
```

Fig.3. Subprogram de eşantionare şi memorare

Prin conectarea la intrarea de semnal analogic a unui amplificator de microfon , iar la ieşire a unui amplificator audio,utilizînd subrutinele prezentate,a fost posibilă înregistrarea şi redarea, cu performanţe acceptabile , a semnalelor din spectrul vorbirii.

```
        DI
        LD      A, 80H
        OUT     (FF), A
        LD      HL, ADR
        LD      DE, NROCT
L2 :    LD      B, INT
        LD      A, (HL)
        OUT     (9F), A
        INC     HL
L1 :    DJNZ    L1
        DEC     DE
        LD      A, D
        OR      E
        JR      NZ, L2
        EI
        RET
```

Fig.4. Subprogram de reproducere a semnalului memorat

O altă aplicaţie interesantă , în care sistemul de achiziţie prezentat îşi dovedeşte eficacitatea, faţă de osciloscopul clasic, constă în memorarea şi reproducerea pe ecranul TV a semnalelor analogice de joasă şi foarte joasă frecvenţă, ce ar putea constitui răspunsul unor sisteme la diferite regimuri tranzitorii. Evident, în funcţie de caz, se pot elabora fără dificultate, diferite programe cu formate adecvate de afişare.

4. Concluzii

Sistemul de achiziţii prezentat, avînd la bază un CAN cu urmărire, dat fiind preţul de cost scăzut şi avantajele menţionate, se dovedeşte eficient într-o gamă largă de aplicaţii, cum ar fi achiziţia semnalelor de joasă frecvenţă provenite de la diverse traductoare , eşantionare şi reţinere în mod continuu şi bucle de reglare automată.

Bibliografie

1. " A low cost, high-performance tracking A/D converter", PMI Application Notes AN-6, 1977.

2. M.Crişan, M.Vlăduţiu "Sistem de achiziţie de date interfaţat cu microcalculatorul TIM-S", INF, nr.1, 1987.

AS.ING. TEA TOROCZKAY

MODULE IN LIMBAJUL MICRO-PROLOG

1.Introducere.

Dupa incarcarea interpretorului micro-PROLOG in memorie, utilizatorul se afla in "spatiul de lucru", fapt semnalat prin aparitia pe ecran a cursorului "&.".

Exista posibilitatea de a defini sau de a incarca de pe caseta unul sau mai multe module. Un modul poate fi privit drept o colectie de relatii (predicate) care comunica cu alte programe prin lista de import si lista de export. Constantele, deci si numele de relatii, definite in spatiul de lucru sau module sint proprii spatiului de lucru respectiv modulului in cauza. In spatiul de lucru se pot utiliza constantele definite si specificate in listele de export ale modulelor existente in memorie si invers, modulele pot utiliza constantele definite in spatiul de lucru sau alte module existente in memorie si specificate in lista lor de import. Modulele pot fi sterse din memorie prin comanda KILL<nume-mod>.

2.Module utilitare [1]

Firma producatoare a interpreterului micro-PROLOG a realizat o colectie de programe utilitare sub forma de module. In continuare se prezinta utilizarea modulelor "trace-mod", "spytrace-mod" si "modules-mod".

Aceste module se incarca de pe caseta cu:

LOAD TRACE , LOAD SPYTRACE si LOAD MODULES.

Se pot lista cu:

LIST trace-mod , LIST spytrace-mod , LIST modules.

Listind modulele se poate vedea pentru fiecare lista de import si de export.

2.1.Modulul trace-mod se foloseste pentru depanarea programelor. El exporta numele de relatie **??** care se foloseste similar cu predicatul **?**:

??(atom1 atom2 ... atomk)

Predicatul ?? are drept efect evaluarea secventei specificate urmarind modul de executie intern al interpreterului.

La inceputul evaluarii fiecarei relatii se afiseaza urmatoarele informatii:

- o lista de numere care reprezinta "istoria" relatiei apelate fata de comanda initiala. Lungimea listei este data de "adincimea" evaluarii.
- numele relatiei care se evalueaza urmata de argumentele pentru care se evalueaza in momentul respectiv.

La evaluarea fiecarei relatii apare mesajul **trace?** si se poate raspunde y, n, s, f, q.

La raspunsul n relatia se evalueaza fara explicatii.

La raspunsul y relatia se evalueaza obtinindu-se explicatii asupra modului de rezolvare.

La raspunsul s nu se face nici o evaluare, relatia se considera adevarata.

La raspunsul f nu se face nici o evaluare, relatia se considera ca fiind falsa pentru a forta "backtrackingul".

La raspunsul q se abandoneaza cererea de tracing.

In continuare se prezinta programul de rezolvare al urmatoarei probleme [2]: intr-o colonie de iepuri exista in anul 1985, 500 de iepuri. Daca in fiecare an numarul lor creste cu 15%, atunci se pune intrebarea: citi iepuri vor fi in anul X?

```
&.L    3
((nr_iepuri X)
  (nr X Y)
  (PP In anul X vor fi Y iepuri))
((nr 1985 500))
((nr X Y)
  (dupa_1985 X)
  (anterior Z X)
  (nr Z x)
  (procent Y x)
  (PP Z x))
((dupa_1985 X)
  (LESS 1985 X))
((anterior X Y)
  (SUM X 1 Y))
((procent X Y)
  (TIMES 1.5E-1 Y Z)
  (SUM Y Z X))
```



Programul se lanseaza cu comanda "nr_iepuri 1995" si se obtin urmatoarele rezultate:

```
&.n r _ i e p u r i   1 9 9 0
1985 500
1986 575
1987 6.6125E2
1988 7.604375E2
1989 8.7450312E2
In anul 1990 vor fi 1.0056785E3 iepuri
```

Rezolvarea interna a acestui program se poate urmari cu comanda

```
??((nr_iepuri 1986))
```

si se obtine

```
&.? ? ( ( n r _ i e p u r i   1 9 8 6 ) )
(1) (nr_iepuri 1986) trace?.y
(1) matches clause 1
(1 1) (nr 1986 X) trace?.y
(1 1) matches clause 2
(1 1 1) (dupa_1985 1986) trace?.y
(1 1 1) matches clause 1
(1 1 1 1) (LESS 1985 1986)
(1 1 1 1) solved (LESS 1985 1986)
(1 1 1) solved (dupa_1985 1986)
(2 1 1) (anterior X 1986) trace?.y
(2 1 1) matches clause 1
(1 2 1 1) (SUM X 1 1986)
(1 2 1 1) solved (SUM 1985 1 1986)
(2 1 1) solved (anterior 1985 1986)
(3 1 1) (nr 1985 X) trace?.y
(3 1 1) matches clause 1
(3 1 1) solved (nr 1985 500)
(4 1 1) (procent X 500) trace?.y
(4 1 1) matches clause 1
(1 4 1 1) (TIMES 1.5E-1 500 X)
(1 4 1 1) solved (TIMES 1.5E-1 500 75)
(2 4 1 1) (SUM 500 75 X)
(2 4 1 1) solved (SUM 500 75 575)
(4 1 1) solved (procent 575 500)
(5 1 1) (PP 1985 500)
```

```
1985 500
(5 1 1) solved (PP 1985 500)
(1 1) solved (nr 1986 575)
(2 1) (PP In anul 1986 vor fi 575 iepuri)
In anul 1986 vor fi 575 iepuri
(2 1) solved (PP In anul 1986 vor fi 575 iepuri)
(1) solved (nr_iepuri 1986)
```

2.2.Modulul spytrace-mod permite urmarirea modului de evaluare a predicatelor in mod selectiv si furnizeaza in acelasi timp informatii mai succinte. El exporta relatiile:

spy, unspy, spying.

a) Relatia **spy** poate fi folosita ca si comanda:

spy R

unde R este un nume de relatie. Prin LIST R se poate vedea ca la relatia specificata R se mai adauga o clauza in plus de forma:

((R!X)(spypoints on)(/)spying R X))

b) Relatia **unspy** poate fi folosita ca si comanda

unspy R

unde R este numele unei relatii pentru care s-a executat o comanda spy. Prin LIST R se poate vedea ca se sterge clauza adaugata prin "spy R".

c)1. Relatia **spying** se poate folosi ca si comanda:

spying on avind drept efect introducerea unei noi clauze si anume ((spypoints on)).

spying off avind drept efect stergarea clauzei spypoints.

2. Relatia **spying** se poate folosi ca si relatie sub forma

(spying R(argumente))

Impreuna cu predicatul ? se poate testa comportarea fiecarei relatii dintr-un program, pentru secvente diferite de argumente. De exemplu executati:

?((spying anterior(x 1983))

unde "anterior" este o relatie definita in programul de mai sus.

2.3.Modulul modules-mod se foloseste pentru editarea in spatiul de lucru si gestionarea modulelor. El exporta relatiile: unwrap, wrap si save-mods. La folosirea acestor relatii se utilizeaza caseta pentru crearea unor fisiere intermediare. Inainte de folosirea lui wrap si unwrap in spatiul de lucru se sterg toate clauzele prin KILL ALL.

a) Relatia **unwrap** se poate folosi ca si comanda

unwrap M

unde M este numele unui modul care se sterge, se trece pe caseta si se aduce in spatiul de lucru. De asemenea se adauga in spatiul de lucru relatia:

((Module M <lista export><lista import>))

accesibila, deci permitind modificarea caracteristicilor modulului.

b) Relatia **wrap** se poate folosi ca si comanda

wrap fisier

unde "fisier" reprezinta numele unui fisier pe caseta in care se salveaza continutul spatiului de lucru, organizat sub forma unui modul cu caracteristicile fixate prin relatia **Module**. Modulul creat se poate incarca cu LOAD fisier.

c) Relatia **save-mods** se poate fol: sub forma unei relatii

(save-mods <nume fisier><lista module>)

si are drept efect salvarea in fisierul indicat a modulelor mentionate. De exemplu,

?((save-mods F(trace-mod spytrace-mod)))

3.Module create de catre utilizator

Fiind in spatiul de lucru, modulele existente pe caseta se incarca in memorie cu

LOAD Fisier

unde Fisier contine un modul cu numele fisier-mod. Tot din spatiul de lucru, se pot lista modulele si relatiile exportate de module. De asemenea, modulele existente in memorie se pot salva prin

?((SAVE Fisier fisier-mod))

Utilizatorul creaza si are acces la module prin comenzile:

(CMOD x) care furnizeaza in variabila x numele modulului;

(CRMOD <nume modul><lista import><lista export>) care creaza un modul nou si transfera utilizatorul in spatiul modulului creat si deschis. Aceasta comanda se poate efectua numai din spatiul de lucru, revenirea din modul facindu-se tot acolo. Nu se pot face salturi intre module. Cursorul

afisat intr-un modul este <nume modul>.

(OPMOD <nume modul>) permite accesul in modulul specificat, presupunind ca acesta exista deja in memorie.

(CLMOD) produce revenirea din modulul curent in spatiul de lucru.

In continuare se prezinta citeva exemple de prelucrari de module, stiind ca in memorie sint incarcate modulele trace-mod, spytrace-mod, modules-mod si ed-mod. Acest ultim modul este format din editorul prezentat in [3] si exporta relatiile ED, EDIT si importa constantele M, I, D, E.

Modulul fermier-mod prezinta programul care rezolva urmatoarea problema [2]: un fermier este pe malul sudic al unui riu impreuna cu o capra, o varza si un lup. Vrea sa treaca impreuna cu acestea pe malul nordic al riului folosind o barca cu doua locuri. Se considera ca unul din obiectele posedate de fermier ocupa un loc in barca. Se cere sa se afle solutia problemei sub forma unei secvente de treceri peste riu. Rezultatul problemei se obtine prin executarea comenzii: "solutie x", unde "solutie" este o relatie exportata de catre modul. Rezultatul se obtine sub forma unei liste care se citeste de la dreapta la stinga.

```
&.L I S T   A L L
((ad b1)
  (ADDCL ((b 1) (PP sint in (b 1) clauza 1) (a 1) FAIL)))
&.? ? ( ( C R M O D   t e s t - m o d   ( f
   g ) ( ? ?   E D   M   D   a d   b 1 ) )
)
(1) ((CRMOD test-mod (f g) (?? ED M D ad b1))
(1) solved (CRMOD test-mod (f g) (?? ED M D ad b1))
test-mod.( ( h   1 ) ( a d   b 1 ) )
test-mod.h   1
test-mod.( ( g   1 ) )
test-mod.C L M O D   .
&.L I S T   b
((b 1)
  (PP sint in (b 1) clauza 1)
  (a 1)
  FAIL)
&.L I S T   g
((g 1))
&.L I S T   h
&.h   1
&.O P M O D   t e s t - m o d
test-mod.? ? ( ( h   1 ) )
(1) (h 1) trace?.y
(1) matches clause 1
(1 1) (ad b1) trace?.y
(1 1) matches clause 1
(1 1 1) (ADDCL ((b 1) (PP sint in (b 1) clauza 1) (a 1) FAIL))
(1 1 1) solved (ADDCL ((b 1) (PP sint in (b 1) clauza 1) (a 1) F
AIL))
(1 1) solved (ad b1)
(1) solved (h 1)
test-mod.C L M O D   .
&.K I L L   A L L
&.u n w r p   a p   e d - m o d
Unwrapping ed-mod onto scratch file
Start tape for recording
Hit ENTER key when
 ready ready

Rewind tape to start of saved program
Loading program from tape

ed-mod is now in workspace
&.L I S T   M o d u l e
((Module ed-mod (ED EDIT) ()))
&.E D   M
NUME PREDICAT :
.M o d u l e
NUMAR "CLAUZSTR$ " :
.1
 ((Module ed-mod (ED EDIT) ())) ((Module ed-mod (ED EDIT) (M )))
```

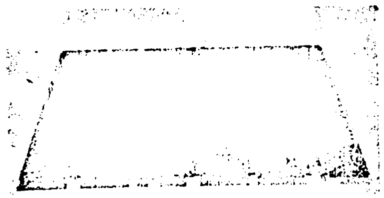
se adauga in lista import constantele M,E,I,D

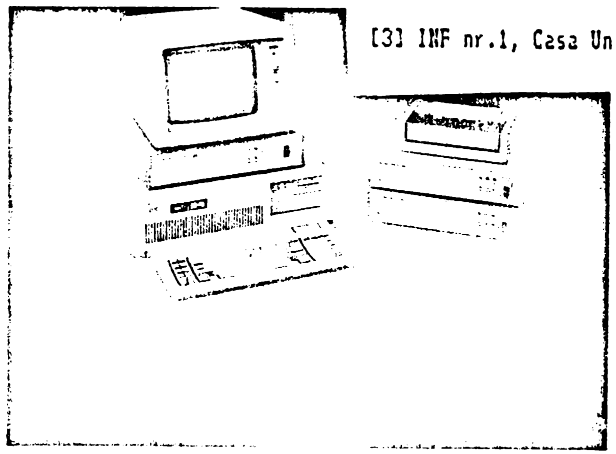
```
 ))) E )))   ))) I )))   ))) D ))) )))
&.w r a p   F E D
Saving module ed-mod in file FED
Start tape for recording
Hit ENTER key when
 ready ready

Workspace is now clear
&.L O A D   F E D

&.O P M O D   e d - m o d
ed-mod.L I S T   D I C T
((DICT ed-mod (ED EDIT) (M E I D) "@MSCRIETI CLAUZA@M" "@MCLAUZA
 S-A STERS@M" CLAUZA NUMAR : PREDICAT NUME KO = LIBER SPATIU))
ed-mod.C L M O D   .
.? ( ( O P M O D   f e r m i e r - m o d (
s o l u t i e ) ( ) ) )
fermier-mod.L O A D   F

fermier-mod.s o l u t i e   x
solutia cautata este : ((f c) (f f) (f l) (f c) (f v) (f f) (f c
))
fermier-mod.C L M O D   .
&.s o l u t i e   x
solutia cautata este : ((f c) (f f) (f l) (f c) (f v) (f f) (f c
))
&.L I S T   f e r m i e r - m o d
((solutie X)
  (actleg X (N N N N))
  (PP solutia cautata este : X))
((actleg () (S S S S)))
((actleg (X|Y) Z)
  (actleg Y x)
  (transforma Y Z x)
  (este_pace Z))
((transforma (f v) (X Y Z X) (x Y Z x))
  (opus X x))
((transforma (f c) (X Y X Z) (x Y x Z))
  (opus X x))
((transforma (f l) (X X Y Z) (x x Y Z))
  (opus X x))
((transforma (f f) (X Y Z x) (y Y Z x))
  (opus X y))
((opus N S))
((opus S N))
((ok_c (X Y X Z)))
((ok_c (X Y Z x))
  (opus Y Z)
  (NOT EQ Z X))
((ok_v (X Y Z X)))
((ok_v (X Y Z x))
  (opus Z x)
  (NOT EQ X x))
((este_pace X)
  (ok_c X)
  (ok_v X))
```





Bibliografie:

[1] Sinclair ZX Spectrum - micro-PROLOG Reference Manual, 1984

[2] Tom Canlon: Start Problem Solving with PROLOG, Addison Wesley 1985

[3] INF nr.1, Casa Universitarilor Timisoara 1988.

MIODRAG PUTERITY
WITTMANN ANTAL

VU-CALC

1. INTRODUCERE

Se incarca si se ruleaza tastind:
LOAD "VU-CALC"

Calculatoarele de buzunar au devenit instrumente puternice si indispensabile pentru multi, daca se lucreaza cu si se afiseaza un singur numar odata. Ele sint utile deoarece multe lucruri din viata de zi cu zi sint descrise bine de un singur numar. Sint insa multe alte lucruri care pot fi bine descrise mai complet cu un tabel sau un tablou de numere organizate in mod ordonat.

VU-CALC e un program pentru calculul si afisarea tabelelor alcatuite din numere si nume. Incepeti cu un tabel gol (o grila compusa din celule aranjate pe linii si coloane). Cu un simplu set de comenzi se pot apela formule de calcul care leaga o celula de alta, o linie de alta sau o coloana de alta, astfel incit computerul poate calcula un intreg tabel in citeva secunde. De asemenea se pot introduce date sau nume in anumite celule, schimba unul sau mai multi parametri, reevalua si afisa tabele pentru diferite situatii aproape instantaneu.

Aceste facilitati ofera un instrument foarte puternic pentru analiza financiara, bugete, calculul tabelelor ingineresti sau stiintifice, analiza statistica etc.

2. FORMAT - TABEL, CURSOR SI FEREASTRA

La intrarea in VU-CALC se poate observa:
- un panou de comanda ; format din doua linii in partea superioara a ecranului.
- o zona goala etichetata ; in mijlocul ecranului.
- o linie de intrare (introducere) ; in partea de jos a ecranului.

Zona goala din mijlocul ecranului e o fereastra pe tabel.

Tabelul poate fi privit ca o multime de celule ordonate in linii si coloane.

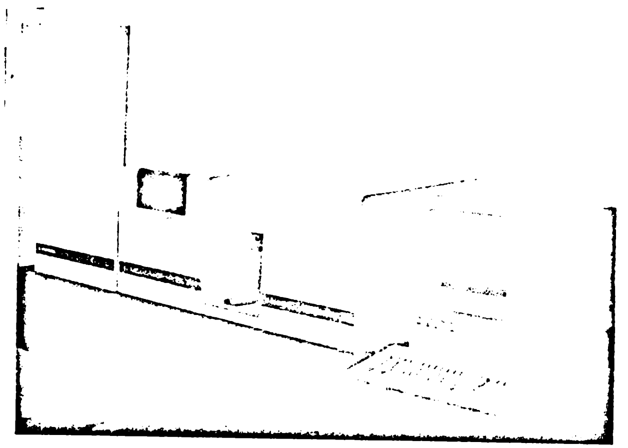
Liniile sint etichetate alfabetic iar liniile adresate curent in fereastra pot fi vazuta in marginea din stinga.

Coloanele sint etichetate numeric de la 01 in ordine crescatoare si coloanele adresate curent in fereastra sint numerotate de-a lungul liniei, imediat deasupra ferestrei.

Fiecare celula e identificata in mod unic de litera corespunzatoare liniei, urmata de numarul coloanei. Astfel A01 sau A1 se refera la celula din coltul din stinga sus al tabelului.

In orice moment atentia utilizatorului VU-CALC este orientata spre o anume celula, pusa in evidenta de un dreptunghi rosu. Acesta este cursorul. Cursorul poate fi deplasat liber si rapid pentru a citi sau introduce date, texte sau formule. Cursorul este deplasat in tabel apasind cele 4 taste cu sageti de pe tastatura (CAPS SHIFT impreuna cu una dintre tastele 5,6,7,8). Cursorul poate fi deplasat rapid mentinind apasata una dintre tastele cu sageti.

Cind cursorul ajunge la o celula din extremitatile ferestrei, fereastra se va muta automat in tabel. Acest lucru poate fi observat remarcind numarul de coloana sau de linie schimbindu-se pe laturile ferestrei. Prin aceasta metoda, utilizatorul se poata deplasa rapid de-a lungul unui tabel foarte mare (in cazul unui Spectrum de 48K).



3. INTRODUCEREA DATELOR SI A TEXTULUI.

Utilizatorul poate privi VU-CALC-ul similar cu un spread-sheet in care-si poate aranja dupa dorinta text si date numerice.

Din comanda principala a VU-CALC-ului se pot introduce 4 tipuri de marimi: text, date, formule sau comenzi.

Pentru a introduce text, se pozitioneaza cursorul in celula de la care se doreste inceperea textului. Se tasteaza " si apoi textul dorit de la respectiva pozitie. Tastind veti observa textul scris la linia de intrare din partea de jos a ecranului cu un mic cursor rosu ce se deplaseaza de-a lungul acesteia. Puteti umple intreaga linie sau puteti folosi tasta <DELETE> pentru a face modificari. Odata terminat apasati <ENTER> pentru a insera textul in spread-sheet.

Pentru a introduce un numar intr-o celula pozitionati cursorul in locul in care doriti sa-l introduceti, tastati numarul urmat de <ENTER>. Numarul va fi afisat instantaneu in celula curenta.

Pentru a calcula un numar intr-o anumita celula folosind o formula, pozitionati cursorul in celula respectiva si tastati formula. Cind formula (afisata in partea de sus a ecranului) e corecta apasati <ENTER> care va aplica formula celulei curente si va calcula datele din celulele aferente. O formula se poate aplica mai multor celule cu comanda # Repeat (de repetare). VU-CALC face distinctie automat intre text, date si formule.

Cind linia de introducere e goala, tastind "#" se va introduce VU-CALC-ul in modul de comanda si in cele doua linii superioare va apare o lista de comenzi. Apasind tasta corespunzatoare primului caracter a comenzii dorite, aceasta va fi executata in functie de parametrii ceruti.

4. FORMULE

Adevarata putere a VU-CALC-ului provine din folosirea formulelor pentru celule, linii sau coloane pentru a genera datele necesare completarii tabelului.

Sintaxa unei formule include folosirea constantelor (numerelor), referinte pentru numere din alte celule si operatori aritmetici simpli: +,-,/,*. Numerele din alte celule sint privite ca referinta celulei pe care o ocupa (litera ce indica linia si numarul coloanei).

In construirea formulelor, referintele la celule trebuie privite ca variabile iar formulele ca expresii algebrice simple care folosesc aceste variabile, constante si operatori aritmetici.

Exemple de astfel de formule sint:

B1*1.03
D12*(B2+1.5)/C1
D7-C7

O formula se poate referi la o celula anume, sau poate fi repetata de-a lungul unei linii, pe o coloana sau de-a lungul unui bloc. Pentru a obtine aceasta se foloseste comanda # Repeat (vezi comenzile de mai jos).

Formulele se aplica intotdeauna relativ. Spre exemplu in comanda de repetare, daca formula curenta e aplicata unei secvente de celule de-a lungul unei linii atunci referinta la calcul sint intotdeauna incrementate, astfel incit formula se aplica secvential de-a lungul liniei. De exemplu, daca formula "1.03*A1" din celula A2 este repetata de-a lungul liniei A, atunci formula din celula A3 va fi "1.03*A2",iar cea din celula A4 va fi "1.03*A3". Acelasi concept relativ e aplicat coborind pe o coloana sau de-a lungul unui bloc unde literele ce identifica liniile sint incrementate secvential.

Daca intr-o formula doriti sa va referiti la o anume celula care nu se modifica cind se repeta, referinta la celula va fi precedata de caracterul "$". Astfel, in exemplul de mai sus, formula "1.03*$A1" aplicata liniei A, se va referi intotdeauna doar la continutul celulei A1.

5. COMENZI

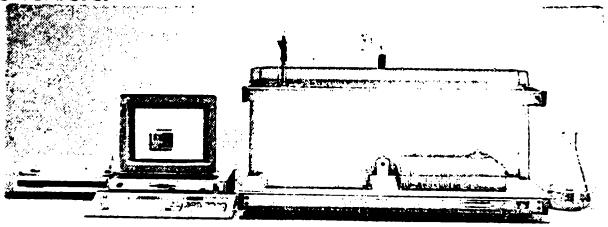
VU-CALC-ul ofera o gama larga de comenzi pentru a incarca, salva si tipari fisiere, pentru editare etc.

Aceste comenzi sint introduse tastind caracterul "#" in linia de intrare, urmat de primul caracter al comenzii.

#B BLANK
Sterge celula curenta.

#C CALCULATE
Forteaza recalcularea intregului tablou fiind necesara uneori dupa modificarea unei formule.

#E EDIT
Permite ca formula din celula curenta sa fie inlocuita cu o alta formula.



#F,c,f,j FORMAT
Aceasta comanda format, specifica reprezentarea unui numar intr-o coloana definita de trei parametri c,f,j.

Primul parametru (c) trebuie sa fie un numar din 1 sau 2 cifre sau litere "A", daca (c) e un numar, formatul se va aplica doar acelei coloane, in timp ce (f) se va aplica intregului tablou.

Parametrul (f) specifica tipul de format dorit. Daca se specifica I, se foloseste forma intreaga. Daca se specifica "$", se foloseste un numar real cu 2 zecimale. G specifica un format general.

Al 3-lea parametru (j) trebuie se fie L sau R dupa cum se doreste alinierea la dreapta sau la stinga.

#G,lc GO
Muta cursorul in celula "lc" (linie-coloana).

#L LOAD
Sterge ecranul si cere un nume de fisier dupa care incarca acel fisier.

#P PRINT
Face ca o copie a ecranului sa fie trimisa la imprimanta.

#Q QUIT
Permite utilizatorului sa stearga pagina de lucru sau sa paraseasca programul.

#R,lc,p:u REPEAT
Continutul celulei "lc" (linie-coloana) este reprodus pe intreg intervalul specificat. Cel mai important fapt e ca va repeta formula din celula "lc" si la celelalte celule din intervalul specificat.

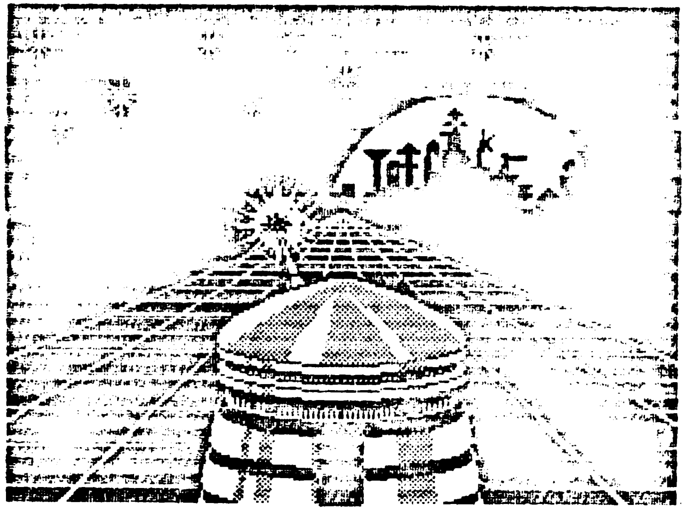
Primul parametru "lc" e o referinta de celula, de exemplu B3, care specifica celula a carui continut sau formula va fi repetata.

Celula poate fi repetata ,de-a lungul liniei, pe coloana sau pe un bloc rectangular de celule. Acesta e definit de domeniul parametrilor de forma p:u, unde p si u sint referintele la celule ,de exemplu A5:B5; p este celula din stinga-sus a blocului iar u este celula din coltul din dreapta-jos a blocului. In exemplul de mai sus, celulele din llcc vor fi repetate in celulele A3,A4,A5,B3,B4,B5 din domeniu. Celula p trebuie sa fie intotdeauna in stinga si deasupra celulei u.

#S SAVE
Sterge ecranul si cere un nume de fisier. Datele din tabel vor fi salvate pe caseta.

#T,l sau c,l'sau c' TRANSFER

Aceasta comanda transfera o linie sau o coloana definita de primul parametru pe alta linie sau coloana definita de al doi-lea parametru. O linie nu poate fi copiata pe o coloana si viceversa.



6. FACILITATI DE SUMA

O parte dintr-o linie,dintr-o coloana sau dintr-un bloc dreptunghiular poate fi adunata automat cu ajutorul facilitatii de suma (adunare). Adunarea e tratata ca o formula. Se pozitioneaza cursorul in celula unde se doreste plasarea rezultatului sumei. Se introduce o formula de forma: & p:u si se apasa tasta <ENTER>. Simbolul "&" inseamna suma, iar formula de mai sus semnifica "aduna celulele incepind cu prima celula p pina la ultima celula u".

p:u e un domeniu dreptunghiular pe care are loc adunarea, unde p este celula din stinga-sus a dreptunghiului .

De exemplu: & A2:B4 va aduna celulele A2+A3+A4+B2+B3+B4. Pentru a aduna linia C de la C3 la C10 se introduce: & C3:C10.

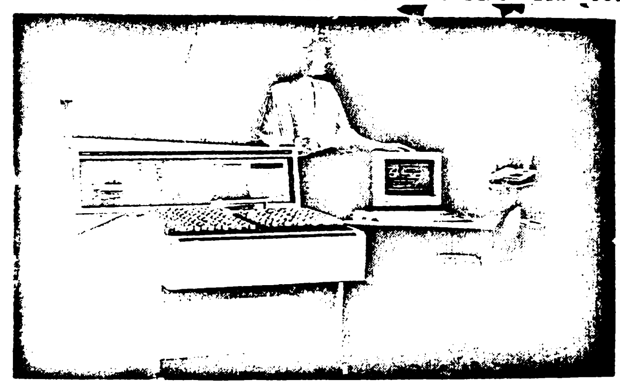
Pentru a aduna coloana 5 de la B5 la D5 se introduce & B5:D5.

Celula p trebuie sa fie intodeauna deasupra si in stinga celulei u.

7. ERORI

Daca se introduce o formula care se refera la o celula goala sau care contine caractere in loc de date numerice, pot apare erori, iar programul va cadea cu un cod de eroare a Spectrumului afisat in partea de jos a ecranului.

Daca se intimpla asa ceva programul poate fi reinceput introducind GO TO 9000. Programul va afisa identificatorul celulei care a produs eroarea. Apasati <ENTER>. Pe ecran se va reveni la tabel si se poate muta cursorul in celula cu probleme. Priviti formula din aceasta celula si vedeti care din celulele



la care se refera e goala sau contine caractere nenumerice.

8.) SUMARUL COMENZILOR VU-CALC.

- #B - sterge celula curenta si formula sa.
- #C - forteaza recalcularea intregului tabel - necesara cind se schimba o data sau o formula.
- #E - pentru schimbarea formulei din celula curenta.
- #F,c,f,j - formateaza celula c (A=All (toate)). f=I (intreg), =$(zecimal), =G(general). j (aliniat la) =L (left (stinga)), =R (right (dreapta)).
- #G,lc - pozitionare pe celula lc.
- #L - incarca un fisier de date VU-CALC.
- #P - tipareste o copie a ecranului.
- #Q - paraseste VU-CALC-ul.
- #R,lc,p:u - repeta continutul celulei lc pe domeniul celulelorde la p la u.
- #S - salveaza fisierul curent VU-CALC.
- #T,l sau c,l' sau c' - transfera linia l in linia l' sau coloana c in coloana c'.

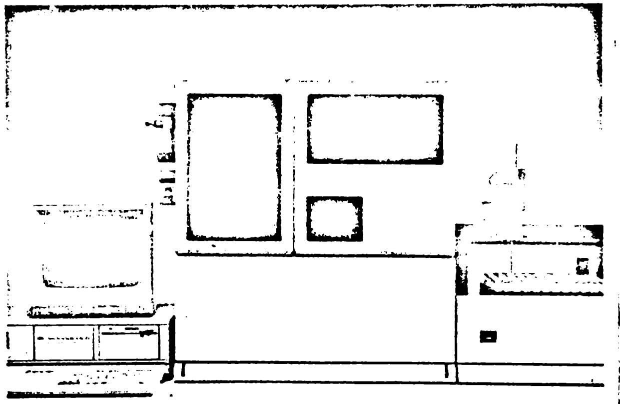
MIDRAG PUTERITY

BLAST COMPILER V3.0

COPYRIGHT

DISCLAIMER

CUPRINS



1. INTRODUCERE

BLAST este primul compilator perfect compatibil BASIC cu posibilitati de optimizare aparut pentru calculatoarele Sinclair. Scopul sau primordial este de a obtine viteza maxima de executie pentru programe scrise in BASIC-ul Spectrum-ului fara a produce coduri obiect de dimensiuni prohibitive. BLAST-ul poate creaste viteza BASIC-ului de pina la 40 de ori.

Operarea cu compilatorul este extrem de simpla. Sint foarte putine comenzi noi care trebuie invatate, iar nivelul de compatibilitate cu interpretorul BASIC atit de inalt incit chiar si extensii ale BASIC-ului scrise in cod masina vor fi compilate. Utilizatorii care scriu specific pentru BLAST pot beneficia de o gama larga de extensii prevazute in compilator, precum si de puternicul toolkit livrat impreuna cu compilatorul.

BLAST a fost dezvoltat de catre aceeasi companie care a produs PETSPEED, compilatorul BASIC pentru Comodore 64 si numeroase alte compilatoare ce ruleaza pe diverse microcomputere. In afara BLAST-ului compania produce si OXFORD PASCAL, o implementare completa a popularului limbaj Pascal pentru diverse microcomputere incluzind Spectrum-ul.

Speram ca BLAST-ul sa va placa si sa-l gasiti folositor. Daca aveti comentarii sau (oare sa spunem ?) gasiti bug-uri (erori de programare) am fi deosebit de multumiti sa le auzim de la d-voastra. Va rugam sa ne scrieti in loc sa ne telefonati, deoarece datorita pretului scazut al BLAST-ului, service-ul prin telefon nu este posibil.

2. DESPRE ACEST MANUAL

Oricine are un program BASIC caruia trebuie sa i se mareasca viteza, poate beneficia de BLAST. Folosirea BLAST-ului insusi, nu necesita nici o cunostinta despre BASIC. Deoarece in principal utilizatorii vor fi programatori in BASIC, acest manual li se adreseaza in mod special.

Manualul BLAST-ului nu incearca sa-i invete pe incepatori programarea in BASIC, ci doar discuta despre programare cind aceasta este necesar pentru a face referiri la compilator.

BLAST-ul este atit de simplu de utilizat incit utilizatorul este tentat sa treaca prea rapid prin manual sau chiar sa-l ignore complet. Noi descurajam in mod hotarit aceasta tendinta. Utilizatorii sint sfatuiti sa citeasca manualul atent inainte de a incepe orice lucru mai serios cu compilatorul.

3. CE ESTE UN COMPILATOR ?

In aceast capitol vom explica notiuni de baza despre compilatoare si citeva elemente de terminologie. O intelegere a conceptelor de baza prezentate aici, altfel neesentiala, va va sporii mult abilitatea in exploatarea BLAST-ului. Scurta sectiune de terminologie trebuie citita si inteleasa.

Un program BASIC este pur si simplu o portiune de text in care specificam actiunile pe care vrem sa le intreprinda computerul cind programul ruleaza. Microprocesorul Z80 cu care este echipat Spectrum-ul intelege doar un limbaj numit cod masina. Pentru Z80, un program BASIC este o intreaga babilonie. Pentru a rula un program avem nevoie de software care sa inteleaga BASIC-ul si sa-l traduca intr-o forma pe care Z80 sa o poata intelege. Exista doua tipuri de programe de traducere a BASIC-ului, interpretorul BASIC si compilatorul BASIC.

1) INTERPRETOARE

Un interpretor BASIC, cum este cel furnizat in ROM ca parte a Spectrum-ului, citeste fiecare instructiune (declaratie) a programului si facind aceasta, intreprinde actiunile

specificate. Interpretoarele sint extrem de utile pentru dezvoltarea programelor, deoarece se interpreteaza textul BASIC propriu-zis. Se poate edita un program, rula si apoi reedita rapid si fara multa bataie de cap. Dezavantajul unui interpretor este acela ca ruleaza incet deoarece aproape tot timpul interpretorul incearca sa inteleaga BASIC-ul in loc sa intreprinda actiunile specificate.

2) COMPILATOARE

Spre deosebire de interpretoare, un compilator traduce intreg programul in ceea ce masina poate intelege, intr-o singura operatie, numita compilare. Cind operatia este terminata, avem un bloc de cod masina care este versiunea tradusa a textului BASIC. Compilatoarele sint mult mai putin utile in stadiul de dezvoltare a unui program, fata de interpretoare, deoarece o schimbare cit de mica in textul BASIC necesita o recompilare completa a programului. Cu toate acestea, odata ce un program a fost compilat, va rula la o viteza mult mai mare.

4. TERMINOLOGIE

In continuare se vor adopta urmatorii termeni:

COMPILE TIME (compilare) - perioada de timp in care BLAST-ul compileaza un program
RUN TIME (rulare) - perioda de timp in care programul compilat este executat
SOURCE FILE (fisier sursa) - fisierul de intrare intr-un compilator, in acest caz text BASIC, uneori numit si cod sursa
OBJECT FILE (fisier obiect) - iesirea dintr-un compilator, in acest caz traducerea in cod masina a unui text BASIC, uneori numit si cod obiect
MACHINE CODE (cod masina) - limbajul intern inteles de microprocesorul Z80
P-CODE (cod "p") - o reprezentare intermediara a unui program, intre BASIC si cod masina, fiind o alternativa a codului masina care necesita mult mai putin spatiu dar si un miniinterpretor la rulare, si este putin mai lenta decit codul masina dar mult mai rapid decit BASIC-ul interpretat ; BLAST-ul poate compila in p-code, in cod masina sau intr-un amestec al acestora
COMPILER DIRECTIVE (directive ale compilatorului) - un mesaj adresat compilatorulu care se adauga textului unui fisier sursa si care afecteaza modul in care se comporta calculatorul; BLAST-ul prezinta o suma de directive compilator foarte utile care se adauga programului sub forma unor instructiuni REM speciale

5. SA INCEPEM

Aceast capitol explica cum se utilizeaza BLAST-ul in modul cel mai simplu. Vom lua un program BASIC deja incarcat in calculator si il vom compila direct in RAM fara acces la banda sau la microdrive. In acest mod (numit mod RAM to RAM) sintem limitati la programe relativ scurte deoarece atit compilatorul cit si programele sursa si obiect trbuie sa coexiste in memorie.

1) Incarcati BLAST-ul dupa cum urmeaza:

LOAD "BLAST" <ENTER>

BLAST-ul va autorula si va "semna" cu mesajul:

BLAST (c) OCSS 1985 xxxx BYTES FREE

In acest punct BLAST-ul va face o verificare de protectie pentru a stabili daca sinteti un utilizator autorizat al acestui produs software. Speram ca nu veti gasi procedura prea obositoare. Secventa de protectie apare doar cind BLAST-ul se incarca pentru prima data. Odata verificarea facuta, BLAST-ul va va permite sa compilati oricite programe doriti, fara alte complicatii.

Intre copertile acestui manual sau pe o foaie de hirtie separata veti gasi o matrice de patratele colorate. Fiecare patratel poate fi identificat printr-o simpla referinta la grila. De exemplu pentru a gasi patratelul E-13, se identifica coloana E si rindul 13 iar patratelul E-13 este cel in care se intersecteaza coloana E si rindul 13 (vezi figura).

```
.  . . . . . . . .
13 . . . . * . .
.  . . . . . . .
.  . . . . . . .
.  . . . . . . .
.  . . . . . . .
1  . . . . . . .
   A B C D E F .
```

Verificarea de protectie este foarte simpla. Trebuie doar sa identificati corect culorile respective. BLAST-ul va da instructiuni dupa cum urmeaza:

ENTER THE COLOUR IN SQUARE x-xx (W,Y,G,R) ?

unde x-xx este o referinta la grila. Cind ati gasit patratelul introduceti una din literele W,Y,G sau R dupa cum patratelul este alb,galben,verde sau rosu si apasati tasta ENTER. Cind ati raspuns corect la patru astfel de intrebari, verificarea de protectie ete terminata.

BLAST-ul este acum complet initializat si gata sa compilze programe. De acum inainte, pina cind se tasteaza NEW sau se debranseaza alimentarea calculatorului, Spectrum-ul va raspunde la un set de comenzi aditionale folosite pentru a comunica cu BLAST-ul. Comenzile BLAST-ului sint precedate de un asterisc (*) pentru a le distinge de cele normale ale Spectrum-ului.

Dorim sa folosim BLAST-ul pentru a compila un program deja existent in memorie si sa avem codul obiect rezultat tot in memorie. In acest mod BLAST-ul se comporta in mod implicit.

Se incarca (sau se tasteaza) un program BASIC, nu mai mare de 5K si se tasteaza:

*C

pentru a-l compila.

In acest moment compilatorul va putea decide daca are nevoie de memoria ecran a Spectrum-ului ca spatiu de lucru. Nu va alarmati pentru orice babilonie care ar putea sa apara pe ecran; este pur si simplu BLAST-ul folosind optim memoria disponibila. Presupunind ca nu sint probleme, dupa un minut sau doua, controlul va va fi reda cu mesajul:

(0) WARNINGS (0) ERRORS

Pentru a rula veriunea compilata a programului, tastati:
*R

Daca BLAST-ul ramine fara spatiu de lucru la compilare, va va intreba daca poate sa stearga din memorie programul sursa. Daca nu doriti ca acest lucru sa se intimple, tastati N si veti reveni la interpretor. In caz contrar, tastati Y si compilarea va contina. BLAST-ul nu va sterge niciodata un program fara permisiune.

Atita timp cit BLAST-ul este in memorie, veti putea edita codul sursa, rula sub interpretor sau compila si rula programul BLAST-at orideciteori doriti. Lucrind cu BLAST-ul veti considera ca uneori trebuie sters programul din memorie fara a sterge BLAST-ul. Pentru aceasta in loc de NEW (care ar sterge toata memoria, inclusiv BLAST-ul), folositi directiva *N. Aceasta sterge orice text BASIC fara a afecta compilatorul.

N.B. Desi BLAST-ul poate face fata la codul masina scris de utilizator si chemat dintr-un program BASIC, acest lucru nu e posibil in modul RAM to RAM (vezi capitolul BLAST SI CODUL MASINA AL UTILIZATORULUI).

6. SALVAREA PROGRAMELOR BLASTATE

Pentru salvarea codului obiect se da directiva *S. BLAST-ul va intreba daca doriti o salvare pe banda sau pe microdrive iar apoi va va intreba numele de fisier sub care doriti sa salvati programul BLAST-at. Bineinteles, se poate utiliza orice nume de fisier legal dar o practica utila ar fi sa folositi numele original al programului BASIC cu un amendament subscris. Fisierul astfel scris pe banda sau pe microdrive contine codul obiect al programului d-voastra impreuna cu sitemul run-time al BLAST-ului. El nu contine nici o parte a BLAST-ului propriu zis.

Tastati numele de fisier si apoi ENTER.

Pentru a verifica daca codul a fost salvat corect, tastati NEW pentru a "goli" calculatorul. Aveti posibilitatea de a incarca codul obiect salvat la fel ca si un program BASIC obisnuit. Pentru a-l executa tastati RUN. BLAST-ul salveaza intotdeauna un cod obiect in asa fel incit se incarca in zona de memorie rezervata fisierului text BASIC. Compilatorul face acest lucru pentru ca programele BLAST-ate sa poata fi incarcate si rulate la fel ca programele BASIC. Deoarece computerul a fost golit, va fi necesar sa se incarce din nou BLAST-ul pentru a putea continua.

7. BLAST-AREA PROGRAMELOR LUNGI

Pina acum am vazut cum BLAST-ul compileaza din memorie in memorie. Dupa cum am explicat inainte, acest lucru este posibil daca programul compilat este scurt. Pentru a rezolva aceasta problema, BLAST-ul este prevazut cu optiuni de citire a codului sursa de pe banda sau de pe microdrive, si scriere a codului obiect rezultat pe oricare din aceste periferice. In continuare vom explica cum se foloseste BLAST-ul in diverse moduri de intrare/iesire.

Selectarea dispozitivelor de intrare/iesire

Pentru a selecta dispozitivul de la care BLAST-ul va citi fisierul sursa se foloseste optiunea INPUT tastind:
*I
si raspunzind intrebarii:

ACCEPT INPUT FROM: RAM, TAPE, MICRODRIVE

cu R, T sau M.

Pentru a selecta dispozitivul la care BLAST-ul va scrie codul obiect, se tasteaza:

*O

si se procedeaza ca mai sus.

Oriciteori e selectata optiunea de compilare cu:

*C

BLAST-ul va cere informatii corespunzatoare optiunilor de intrare/iesire alese. Spre exemplu, daca a fost selectat un microdrive, BLAST-ul va cere numarul microdrive-ului si numele de fisier. Daca s-a selectat banda, BLAST-ul va cere doar numele de fisier.

Diferitele combinatii ale dispozitivelor de intrare/iesire, lasa BLAST-ul cu mai mult sau mai putin spatiu de lucru pe perioada compilarii. Daca programul de compilat depaseste aproximativ 5K, probabil va fi necesar ca citirea sa se faca de pe banda sau microdrive, in loc de memorie. Daca programul sursa e foarte lung, (mai mult de 8K) va fi necesara si selectarea iesirii pe banda sau pe microdrive. Insa oricare ar fi dispozitivele selectate, BLAST-ul va va conduce pas cu pas pe parcursul procesului de compilare.

Daca dispozitivul de iesire este banda sau microdrive-ul, compilarea se va sfirsi cu codul obiect inscris pe dispozitivul respectiv. Evident, programul obiect va trebui incarcat de pe suportul sau pentru a putea fi rulat. Retineti ca BLAST-ul insusi consuma o mare cantidate din memoria Spectrum-ului, insa acest lucru nu va impiedica sa compilati programe lungi, ci doar va determina sa inlaturati BLAST-ul din memorie atunci cind rulati astfel de programe. Inlaturarea BLAST-ului din memorie se face cu comanda:

*Q

BLAST-area pe microdrive

Este cel mai bun mod de a compila programe lungi. Daca aveti astfel de programe de compilat si nu aveti microdrive va recomandam sa va procurati de urgenta unul. BLAST-ul poate fi copiat pe microdrive cu comanda:

*B

BLAST-area pe banda

Daca nu aveti microdrive si doriti sa compilati programe lungi, veti proceda in felul urmator: Datorita naturii limitate a benzii ca dispozitiv I/O, programul de compilat trebuie intii salvat pe banda intr-un format special. Facilitatile necesare sint incluse in TOOLKIT-ul furnizat pe fata opusa a casetei BLAST-ului. Pentru mai multe detalii va trebui sa consultati capitolul dedicat TOOLKIT-ului. Odata ce programul sursa a fost salvat pe banda intr-un format adecvat, BLAST-ul poate fi incarcat in memorie si se poate incepe compilare. Procesul e continuu in cazul in care programul e suficient de scurt pentru a genera un cod obiect compilat in memorie. Daca programul e mai lung decit 8K, acest lucru nu va fi posibil si va trebui sa folositi banda atit ca dispozitiv de intrare cit si de iesire. Desi BLAST-ul permite acest lucru, calea de urmat e destul de anevoioasa. Daca obisnuiti sa folositi BLAST-ul pentru a compila programe lungi, aveti nevoie de un microdrive.

BLAST-area programelor de pe banda pe banda

In acest mod se folosesc doua benzi: o banda sursa si o banda obiect. Banda sursa contine programul d-voastra in forma speciala necesara compilarii (vezi aliniatul precedent), iar banda obiect e goala.

Sa presupunem ca ati selectat banda atit pentru intrare cit si pentru iesire. Cind tastati *C pentru a incepe compilarea, BLAST-ul va va instrui sa inserati banda sursa si sa apasati pe PLAY. Dupa putin timp calculatorul va fluiera (beep) si va va cere sa schimbati benzile. Banda sursa trebuie oprita in mai putin de 5 secunde de la beep. Daca nu veti proceda astfel, datele urmatoare vor fi pierdute.

Dupa un timp vi se va cere sa schimbati din nou benzile. Timpul de inlocuire a benzii obiect cu cea sursa nu e critic dar va sfatuim sa fiti rapid(a) pe perioada intregului proces pentru a micsora timpul total de incarcare a programului BLAST-at.

Pe durata compilarii vi se va cere sa schimbati benzile intre ele de un numar de ori ce depinde de lungimea programului ce se compileaza. In final, compilatorul va afisa obisnuitul raport despre starea erorilor.

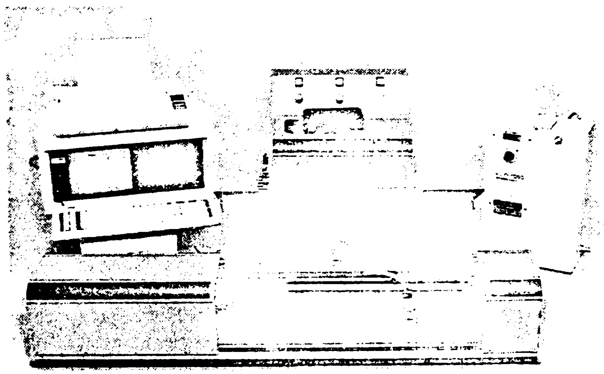
Cind compilati de pe banda pe banda, fisierul obiect este scris intr-un format oarecum nestandard. Cu toate acestea, in afara de faptul ca timpul de incarcare este mai lung decit in mod normal, nu vor exista diferente majore fata de un program obisnuit.

8. P-CODE SI COD MASINA

BLAST-ul poate compila programe atit in cod masina Z80 cit si intr-un pseudo cod masina mai compact, numit p-code. Argumentele pro si contra ale acestor doua tipuri de cod obiect pot fi rezumate in tabelul de mai jos:

	P-CODE	COD MASINA
VITEZA	Mai mare decit a BASIC-ului dar mai mica decit a codului masina.	Cea mai mare posibila.
DIMENSIUNE	Mai mica decit in BASIC mai mica decit in cod masina.	De regula mai mare decit BASIC-ul. Intotdeauna mai mare decit a p-code-ului.

Harta memoriei pentru un program BLAST-at este data in ANEXA 1. Se observa ca in afara de cod obiect si date, un program BLAST-at contine si un bloc de cod denumit Run Time System (RTS). RTS-ul este in principal o biblioteca de subrutine



chemate din codul obiect pentru operatii cum ar fi inmultirea, impartirea si manipularea sirurilor. RTS-ul se include intotdeauna intr-un program BLAST-at si necesita in plus 5K din memorie. Din aceasta cauza, un program BLAST-at va fi intotdeauna mai lung decit 5K. Cu toate acestea, avind in vedere ca p-code-ul are cam 2/3 din lungimea echivalentului sau BASIC, programele lungi compilate in p-code pot deveni mai scurte decit originalul. Spre exemplu, un program de 3K dupa ce a fost BLAST-at in p-code va ocupa aproximativ 7K; 2/3*3 pentru p-code si 5K pentru RTS. Similar, un program BASIC de 30K va deveni dupa compilare de 25K. Evident undeva exista un punct de intilnire, la care cele doua marimi sint aproximativ egale. Acest lucru se intimpla la aproximativ 15K.

Evident, cele de mai sus au un caracter foarte general. Unele tipuri de programe genereaza mai putin p-code decit altele iar programele care contin numeroase comentarii vor suferi o reducere in lungime mult mai pronuntata decit cele care nu le contin.

Daca BLAST-ul este determinat sa genereze cod masina (in loc de p-code) programul va avea aproape in mod sigur o crestere in lungime. Un astfel de program va rula mai rapid dar acest lucru nu e de folos daca nu va incapea in memorie. Din fericire insa, BLAST-ul poate fi determinat sa genereze cod masina pentru acele sectiuni din program in care viteza e critica, si p-code in rest. De multe ori, compilind o sectiune relativ scurta de BASIC in cod masina si restul in p-code vom avea aproape aceeasi viteza ca si cind programul ar fi fost compilat in intregime in cod masina.

Tipul de cod obiect generat de BLAST este specificat prin intermediul directivelor compilatorului (vezi capitolul referitor la acest subiect).

Pentru a instrui compilatorul sa genereze p-code vom scrie:

```
REM! P-CODE
```

iar pentru a-l determina sa genereze cod masina:

```
REM! MACHINE CODE
```

BLAST-ul genereaza implicit p-code.

9. BLAST SI CODUL MASINA AL UTILIZATORULUI

BLAST-area unui program care cheama subrutine in cod masina nu ar trebui sa prezinte probleme. BLAST-ul a fost proiectat pentru compatibilitate maxima cu BASIC-ul Spectrum-ului si aceasta compatibilitate se extinde la variabile si la formatul de inmagazinare a programului. In particular, au fost prevazute urmatoarele practici adoptate uzual de utilizatorii de Spectrum:

1) Un program BLAST-at poate sa rezerve spatiu pentru cod masina coborind RAMTOP-ul in mod obisnuit.

2) BLAST-ul inmagazineaza variabilele in exact acelasi mod ca si BASIC-ul Spectrum-ului. In consecinta codul masina care ar preluа si manipula variabile poate functiona sub BLAST.

3) Subrutinele cod masina care extind BASIC-ul interceptind subrutina de tratare a erorii din sistemul de operare al Spectrum-ului (sau prin alte metode) vor functiona. Explicatia

acestui fapt surprinzator e urmatoarea: cind la compilare BLAST-ul intilneste o instructiune care apare incorecta sintactic, compilatorul va copia textul suparator in fisierul obiect precedat de un cod special de ESCAPE. La rulare, cind RTS-ul intilneste acest cod de escape, va chema interpretorul BASIC pentru a-l manipula. Daca textul e o eroare de sintaxa veritabila, interpretorul va raporta acest fapt si va reactiona in mod obisnuit. Daca insa textul e o extensie a BASIC-ului care a fost prevazuta, interpretorul se va comporta intocmai ca si cu programul ne-BLAST-at. (N.T. Desi apetisanta, aceasta facilitate nu functioneaza pentru cea mai raspindita extensie de BASIC, Beta Basic.)

Extensiile de BASIC prevazute de BLAST precum si directivele compilatorului se introduc sub forma unor instructiuni REM speciale, recunoscute de BLAST la compilare. Este posibil ca in viitor alte programe comerciale sau chiar codul masina al utilizatorului sa foloseasca aceeasi tehnica pentru a introduce noi comenzi in BASIC. Din acest motiv s-a introdus in BLAST facilitatea de a permite instructiunilor REM sa fie trecute interpretorului daca incep cu caracterul escape %. Daca BLAST-ul intilneste o instructiune REM care incepe cu acest caracter, va genera un cod care va face ca instructiunea REM cu caracterul % indepartat sa fie trecuta interpretorului la rulare. BLAST-ul indica acest lucru cu mesajul:

COMMENT TRANSFERRED AT LINE xxxx

Este posibil ca unele practici obscure sa creeze probleme. Spre exemplu, codul masina continut in instructiuni REM, nu va functiona cu siguranta atunci cind programul e compilat deoarece aceasta metoda de inmagazinare a subrutinelor cod masina depinde de modul in care textul BSIC e organizat in memorie.

N.B. Din cauza unor posibile suprapuneri intre BLAST si codul masina al utilizatorului, compilatorul nu va permite programelor sa cheme subrutine Z80 cind compilarea se face in RAM. Astfel de programe trebuie compilate folosind banda sau microdrive-ul ca mediu de iesire.

10. FOLOSIREA VARIABILELOR INTREGI

Adeseori este posibil sa ajutati BLAST-ul la crearea unui cod mai eficient informindu-l despre orice variabila care va lua valori intregi intre -65535 si +65535. Majoritatea programelor contin multe astfel de variabile si merita osteneala sa informati compilatorul. Variabilele intregi se declara printr-o directiva compilator de forma:

REM! INT <lista de variabile>

de exemplu:

REM! INT I,J,K,A(10,5)

declara variabilele I, J, K si tabloul A(10,5) ca intregi. Declaratia care va initializa variabilele declarate la valoarea 0, trebuie sa apara la inceputul programului, inainte ca acestea sa fie utilizate. In exemplul de mai sus, declaratia A(10,5) serveste ca instructiune de tip DIM pentru respectivul tablou si va inlocui orice instructiune DIM existenta.

Retineti ca daca unei variabile care a fost declarata intreaga i se asigneaza o valoare neintreaga sau in afara domeniului de valori, rezultatele vor fi imprevizibile.

11. COMPATIBILITATEA CU BASIC-UL SPECTRUM-ULUI

BLAST-ul a fost proiectat pentru o compatibilitate maxima cu BASIC-ul Spectrum-ului. Aceasta compatibilitate se extinde nu numai la limbajul propriu-zis ci si la mediu de programare.

In BASIC, este posibila oprirea unui program in timp ce el ruleaza, se pot citi variabile, executa instructiuni s.a.m.d. Rularea poate fi apoi continuata sau reinceputa. Aceste actiuni sint posibile si sub BLAST cu o singura diferenta. Instructiunea CONTINUE nu va functiona cu un program BLAST-at.

12. PROTECTIA PROGRAMELOR BLAST-ATE

BLAST-ul prezinta un numar de mijloace care pot fi folosite pentru a preveni amestecul neautorizat in programele compilate.

1) AUTORUN (autorulare)
Daca directiva compilator

REM! AUTORUN

este inclusa la inceputul unui program BASIC, BLAST-ul va face ca fisierul compilat sa ruleze automat la rulare. Posibilitatea de AUTORUN face pirateria mult mai dificila si duce la un produs mai profesional.

2) P-CODE sigur

Majoritatea programelor disponibile comercial contin subrutine, scrise de regula in cod masina, care verifica daca nu "s-a umblat" prin program si ofera in plus si alte mijloace de

protectie. Deoarece codul Z80 e bine cunoscut celor care se ocupa cu pirateria, aceste subrutine sint adesea gasite si dezactivate. P-code-ul generat de BLAST e un limbaj nedocumentat si astfel ofera un nivel de securitate mult mai inalt decit codul masina. Astfel, se recomanda ca subrutinele de protectie sa se scrie in BASIC si sa se compileze in p-code.

13. COPIEREA PROGRAMELOR BLAST-ATE

Programele compilate nu pot fi salvate direct utilizind SAVE. Comanda:

*S

nu va resalva un program BLAST-at care a fost incarcat de pe banda sau microdrive. Daca doriti copierea unui program care a fost compilat pe unul dintre aceste dispozitive, procedati dupa cum urmeaza:

Salvarea pe banda

1) Incarcati programul BLAST-at in calculator.
2) Inserati urmatoarele linii:

```
15 LOAD "<prog>"
20 RANDOMIZE USR PEEK 23635+256*PEEK 23636+150
```

unde <prog> e noul nume de fisier.

3) Tastati:

```
SAVE "<prog>" LINE 15
```

Puteti verifica (VERIFY) daca codul a fost salvat corect in mod uzual, inlocuind SAVE cu VERIFY in liniile anterioare.

Salvarea pe microdrive

Metoda de salvare pe microdrive este exact aceeasi, cu exceptia ca parametrii de microdrive (cei obisnuiti) trebuie sa fie prezenti. De exemplu pentru a salva programul <prog>, adaugati liniile:

```
15 LOAD *"m";1;"<prog>"
20 RANDOMIZE USR PEEK 23635+256*PEEK 23636+150
```

si tastati:

```
SAVE *"m";1;"<prog>" LINE 15
```

Detalii despre forma exacta in care apar programele BLAST-ate in memorie, sint date in Anexa I.

14. ERORI

1) Erori la compilare.

Desi prin editorul Spectrum-ului nu se introduce decit BASIC corect din punct de vedere sintactic, exista totusi moduri in care se poate impune BLAST-ului un cod sursa incorect. De exemplu, iesirea dintr-un generator de programe (program generator) poate contine erori; de asemenea programul poate fi oricind deteriorat pe banda sau microdrive. In plus exista posibilitatea introducerii de directive compilator eronate sau instructiuni ale unor extensii de BASIC. Din aceste motive, BLAST-ul verifica in mod riguros sintaxa textului care i se ofera.

Totusi, lucrurile nu sint chiar atit de simple. S-ar putea ca o instructiune care apare incorecta pentru BLAST la compilare, sa fie de fapt o extensie de BASIC perfect normala, posibil de genul celor oferite de anumite programe comerciale. Astfel de extensii sint perfect admise sub BLAST, problema fiind doar aceea ca la compilare BLAST-ul nu are suficiente informatii pentru a le distinge de adevaratele erori.

Solutia adoptata de BLAST e urmatoarea: Orideciteori BLAST-ul intilneste o posibila eroare de sintaxa, afiseaza textul "suparator" impreuna cu o avertizare (WARNING). Apoi compilarea continua. Daca se dovedeste la rulare ca a fost de fapt o eroare, rularea se opreste cu mesajul:

NONSENSE IN BASIC

2) Erori la rulare.

La rulare, cu o singura exceptie, programele BLAST-ate vor raspunde cu erori cum ar fi NUMBER TOO BIG sau RETURN WITHOUT GOSUB in exact aceeasi maniera ca si interpretorul. Exceptia se refera la eroarea SUBSCRIPT WRONG. Pentru a evita verificarea continua a indicilor de tablou, la rulare, RTS-ul va ignora aceste erori. Daca indicii ies din domeniu, rezultatele vor fi imprevizibile.

15. DIRECTIVELE COMPILATORULUI

BLAST-ul prezinta anumite optiuni de compilare care pot fi apelate prin directivele compilatorului. Acestea apar in instructiuni REM speciale de forma:

REM! <directiva compilator>

Toate directivele compilatorului sint precedate de REM!. Semnul exclamari (!) permite o cale usoara de a spune daca BLAST-ul sa ignore sau nu textul care urmeaza REM-ului. Exista inca doua tipuri de instructiuni REM speciale recunoscute de BLAST:

REM%

face ca textul comentariului sa fie trecut interpretorului la rulare (vezi BLAST si codul masina al utilizatorului), si:

REM&

folosit ca predecesor pentru instructiunile BASIC permise in plus de BLAST. Acestea sint explicate in capitolul extensiilor de BASIC.

Optiunile compilatorului disponibile sub BLAST sint:

DIRECTIVA	INTELES
1) REM! PCODE	Face ca BLAST-ul sa genereze p-code pina se specifica altceva. Acest mod e implicit.
2) REM! MACHINE CODE	Face ca BLAST-ul sa genereza cod masina pina se specifica altceva.
3) REM! INT I,J,K	Declara variabilele I, J si K ca intregi (vezi capitolul referitor la variabilele intregi).
4) REM! AUTORUN	Face ca programul obiect sa ruleze automat dupa ce s-a incarcat. Aceasta trebuie sa fie prima linie din program.

SUMARUL COMENZILOR

Urmatoarele comenzi sint recunoscute de BLAST in starea sa initiala. Retineti ca o comanda NEW va sterge BLAST-ul complet din memorie. Daca doriti sa stergeti un program BASIC din memorie, folositi:

*N

COMPILE sintaxa *C

Compileaza un program BASIC folosind tipul precedent de periferic de intrare (vezi *I) pentru fisierul sursa si tipul precedent de periferic de iesire (vezi *O) pentru fisierul obiect. Modul implicit pentru intrari si iesiri este RAM.

RUN sintaxa *R

Ruleaza un program compilat. Comanda *R e folosita doar pentru a rula un program compilat din RAM in RAM. Daca a fost selectata banda sau microdrive-ul, programul obiect trebuie incarcat de pe aceste dispozitive si executat cu RUN.

SAVE sintaxa *S

Salveaza un program BLAST-at care a fost compilat in RAM. BLAST-ul va cere detalii despre dispozitiv, numar de microdrive si nume de fisier.

INPUT sintaxa *I

Stabileste dispozitivul de pe care BLAST-ul va citi codul sursa la compilare. BLAST-ul va afisa mesajul:

ACCEPT INPUT FROM: RAM, TAPE, MICRODRIVE

pentru care raspunsul este R, T sau M. Dispozitivul implicit e RAM-ul.

OUTPUT sintaxa *O

Stabileste dispozitivul pe care BLAST-ul va scrie codul obiect la compilare. BLAST-ul va afisa mesajul:

ACCEPT OUTPUT FROM: RAM, TAPE, MICRODRIVE

pentru care raspunsul este R, T sau M. Dispozitivul implicit e RAM-ul.

BACKUP sintaxa *B

Copiaza compilatorul BLAST pe microdrive.

QUIT sintaxa *Q

Paraseste BLAST-ul si elibereaza memoria utilizata de BLAST in favoarea altui cod.

16. EXTENSII DE BASIC

Prezentam in continuare o lista a extensiilor de BASIC recunoscute de BLAST. Deoarece Spectrum-ul nu va accepta text care apare incorect editorului BASIC, toate extensiile sint introduse ca instructiuni REM speciale care incep cu caracterul de escape &.

1) Dezactivarea tastei de BREAK.

sintaxa: REM& BREAK ON
REM& BREAK OFF

Aceste instructiuni activeaza si dezactiveaza tasta de BREAK. Tasta BREAK e activata implicit.

2) WHILE ... WEND

sintaxa: REM& WHILE <conditie>
REM& WEND

Aceasta face ca blocul de instructiuni terminat cu REM& WEND sa fie executat in mod repetat, pina cind (while) <conditie> e adevarata (diferita de 0). Daca <conditie> e falsa la inceput, instructiunile se ignora (se "depasesc").

3) REPEAT ... UNTIL

sintaxa: REM& REPEAT
REM& UNTIL <conditie>

Blocul de instructiuni dintre REM& REPEAT si REM& UNTIL se repeta pina cind (until) <conditie> ce urmeaza lui REM& UNTIL devine falsa (zero). Indiferent de valoarea <conditie>, instructiunile se executa cel putin o data.

4) DOKE

sintaxa: REM& DOKE <ne>,<ne>

Unde <ne> e o expresie numerica. Acesta este un POKE pe 16 biti. Rezultatul celei de a doua expresii este depus in doua locatii de memorie la adresa data de prima expresie. Datele sin inmagazinate in format LO-HI. Ambele expresii trebuie sa fie in intervalul 0 la 65535.

5) DEEK

sintaxa: REM& DEEK <nv>,<ne>

Unde <nv> e o variabila numerica. Acesta esteun PEEK pe 16 biti. Continutul celor doua locatii de memorie de la adresa data de cel de-al doi-lea parametru se asigneaza variabilei numerice din primul parametru. Deci <nv> devine egal cu PEEK (<ne>)+256*PEEK (<ne>+1).

6) CALL

sintaxa: REM& CALL <ne> !<lista de parametrii>!

Cheama subrutina cod masina de la adresa data de expresia numerica <ne>. Parametrii (optionali), separati prin virgula pot fi variabile numerice, in intervalul 0 la 65535, sau adresa unei variabile numerice exprimate &<nume_de_variabila>. Acesti parametrii sint inmagazinati, in ordine, primul fiind in adresa specificata de IX. De exemplu:

REM& CALL 50000,X,&Y

va avea ca rezultat o chemare a subrutinei cod masina de la aresa 50000. La intrarea in subrutina, intregul X va fi inmagazinat in (IX+0) si (IX+1) si adresa variabilei numerice in (IX+2) si (IX+3).

7) ELSE

sintaxa: REM& ELSE:<lista de instructiuni>

O extensie optionala la IF ... THEN, frecventa in numeroase BASIC-uri. De exemplu:

IF x=0 THEN GOSUB 100: REM& ELSE: GOSUB 200

va avea ca rezultat o chemare la linia 100 daca x este 0 si o chemare la linia 200 daca x este diferit de 0. Instructiunile IF ... THEN ... ELSE nu pot fi imbricate (nested) si orice ELSE trebuie sa apara pe aceeasi linie cu IF-ul aferent. RETINETI CA ELSE TREBUIE URMAT DE "DOUA PUNCTE".

8) Functii "multi line"

In BASIC-ul Spectrum-ului exista posibilitatea definirii si apelarii de functii cu parametrii. Principala limitare a functiilor definite de utilizator e faptul ca ele pot contine doar o singura instructiune care trebuie sa fie o expresie.

BLAST extinde aceasta facilitate, permitind functii pe mai multe linii. Acestea pot fi cel mai bine explicate printr-un exemplu. Sa presupunem ca dorim sa scriem o astfel de functie care sa aiba ca rezultat pe cel mai mare dintre cei doi parametrii de intrare. Vom proceda in felul urmator:

```
1000 REM& DEF M(A,B)
1010 IF A>B THEN LET M=A: REM& ELSE: LET M=B
1020 REM& END PROC
```

Functia poate fi chemata cu instructiunea:

```
100 REM& M(X,Y)
```

Linia 1000 defineste functia M. In linia 1010, M, numele de functie e tratat ca o variabila si e egalat cu cel mai mare dintre A si B. Linia 1020 termina procedura, si reda controlul instructiunii de dupa apelare. Parametrii din definitia de procedura, in acest caz X si Y sint locali pentru procedura. Aceasta inseamna ca parametrii sint necunoscuti in afara procedurii. In plus, daca X si Y sint definiti in afara procedurii sau in alta procedura, ei vor fi tratati ca variabile diferite. O procedura poate avea oricite linii ii sint necesare, dar trebuie terminata cu o instructiune REM& END PROC.

Numele si parametrii procedurilor pot fi formati dintr-o singura litera, optional urmata de semnul $. Functiile multi-line pot fi utilizate recursiv.

17. OPTIMIZARI

BLAST-ul nu traduce pur si simplu instructiunile BASIC in echivalentul lor in cod masina, ci aplica si o gama larga de tehnici de marire a vitezei si compactitatii programului obiect. Autorii BLAST-ului au aderat riguros la vechea maxima a producatorilor de compilatoare:

" Nu lasa pina la rulare ce poti face la compilare ! "

Aceasta se aplica in special la calculul indicilor de tablou. Daca un tablou, sa zicem A(10,10), a fost DIM-ensionat cu constante, BLAST-ul va sti adresa unui element dat A(1,2) (referit cu indici constanti) la compilare. In plus, chiar daca un indice e constant, sa zicem A(I,2), BLAST-ul poate sa perfectioneze codul, facind calcule de indici la compilare. In programe care contin numeroase accese la tablouri, se va observa o crestere de viteza semnificativa.

In evaluarea unei expresii, BLAST-ul va lucra in modul cel mai economic pentru calculul valorii expresiei, fara a retine si manipula valori intermediare inutile.

BLAST-ul poate recunoaste aparitia aceleiasi sub-expresii si daca apare de mai multe ori intr-o expresie sau instructiune. In acest caz el va evalua expresia o singura data si apoi va folosi rezultatul calculat.

Daca cantitatea de memorie permite, BLAST-ul va crea spatiu pentru variabile la compilare, in loc sa lase acest lucru pina la rulare. Spre deosebire de BASIC-ul Spectrum-ului, el va folosi toata memoria disponibila pentru a inmagazina variabile inainte de a fi fortat sa consume timp " colectind deseurile ".

In multe cazuri BLAST-ul poate mari viteza buclelor FOR-NEXT calculind numarul de ciclari inainte de a fi facute si folosind drept contor un registru al masinii.

BLAST-ul se foloseste pe larg de aritmetica intreaga. Intregii se pot manipula mult mai rapid decit numerele in virgula flotanta si se va obtine o crestere de viteza semnificativa folosindu-i oridecíteori este posibil. Exista optiunea de declarare a unei variabile numerice ca intreg; in acest caz, la rulare, fiecare valoare e inmagazinata in format intreg.

18. SA OBTINEM CIT MAI MULT DE LA BLAST

Spre deosebire de interpretorul BASIC, BLAST-ul nu trebuie sa piarda timpul cautind prin program dupa numere de linie, instructiuni DATA si definitii de functii. El cunoaste adresa tuturor acestor obiecte si le poate referi direct.

Puteti ajuta BLAST-ul foarte mult urmind citeva principii simple care-i vor permite sa faca o cit mai mare parte din munca la compilare si nu la rulare. Veti gasi ca facilitatile oferite de toolkit va vor fi de mare folos.

In particular, evitati instructiunile de tipul:

GOTO <expresie>.

Acestea forteaza BLAST-ul sa intirzie calculul adresei de salt pina la rulare si sa rezerve memorie (pretioasa) pentru lista tuturor numerelor de linie si adresele lor la rulare. Incercati sa inlocuiti o astfel de <expresie> cu un numar de linie propriu zis a carui adresa se poate determina la compilare. Acelasi lucru se aplica la toate celelalte instructiuni ce folosesc numere de linie.

Desi e perfect legal, incercati sa nu intrati sau sa iesiti din bucle FOR-NEXT. Daca faceti aceasta, BLAST-ul nu va putea sa prevada consecintele si nu va aplica una dintre cele mai puternice optimizari de care dispune.

Sterge intervalul specificat.

3) MOVE   sintaxa: *m<interval>,n

Muta intervalul specificat la linia n, stergind liniile originale.

4) RENUMBER sintaxa: *R<interval>,n1,n2

Renumeroteaza intervalul specificat incepind cu n1, cu pasul n2. Valoarea implicita pentru n2 este 10.

22. FUNCTII PENTRU SIRURI

1) FIND   sintaxa: *F<interval>, sir

Cauta in intervalul specificat prima aparitie a sirului. Daca sirul este omis, functia FIND va utiliza ultimul sir introdus.

2) SEARCH & REPLACE sintaxa: *S<interval>,sir1,sir2

Cauta in intervalul specificat dupa sir1 si il inlocuieste cu sir2. Noua linie va fi verificata sintactic. Daca apare o eroare, prima linie care contine eroarea se afiseaza. Toate liniile precedente in care s-a facut modificarea ramin modificate. Delimitatorii intre intervalul specificat si sir1 si intre sir1 si sir2 nu sint in mod necesar virgule; se poate utiliza orice caracter nenumeric.

23. ALTE COMENZI

1) TRACE sintaxa: *Tn

Ruleaza programul incepind de la linia n afisind numarul de linie a instructiunii in executie. Tasta de SPATIU poate fi utilizata pentru a incetinii executia si cea de ENTER pentru a o opri.

2) KILL sintaxa: *K

Sterge toate instructiunile REM din program care nu incep cu &, ! sau %.

3) WRITE sintaxa: *W<interval>,<nume de fisier>

Salveaza intervalul specificat pe caseta sub <nume de fisier> (max. 10 caractere).

4) BLAST SAVE sintaxa: *B<nume de fisier>

Salveaza programul intr-o forma care se preteaza compilarii de pe banda. Programul va fi salvat in blocuri, impreuna cu informatiile pe care BLAST-ul le necesita la compilare.

5) QUIT sintaxa: *Q

Paraseste toolkit-ul.

24. CUM S-A NASCUT BLAST-UL

OCSS e o fabrica de compilatoare. Compania construieste numeroase compilatoare pentru diferite limbaje si masini. Sintem adesea intrebati cum procedam pentru a produce un compilator ca BLAST-ul; cit timp lucram la el, ce limaje folosim s.a.m.d. In acest capitol, incercam sa raspundem citorva din aceste intrebari.

In ce limbaj a fost scris BLAST-ul ?

Raspunsul este ca BLAST-ul nu a fost scris intr-un limbaj anume. Toate compilatoarele OCSS sint generate automat, folosind "instrumente" de generare automata. Un limbaj cum este BASIC-ul, e un limbaj ca oricare altul si astfel poate fi descris prin mijloace gramaticale. De exemplu, putem incepe prin a defini o propozitie in limba engleza scriind:

```
<propozitie>::=<subiect><verb><obiect>;
```

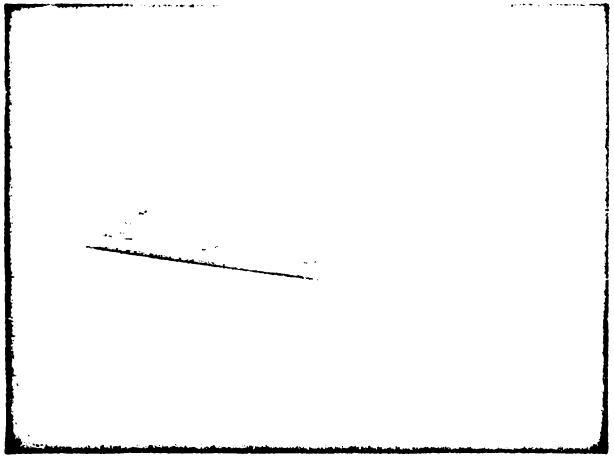
unde ::= inseamna "se defineste ca" iar numele dintre parantezele unghiulare sint obiecte numite "non terminals" care vor fi definite ulterior. De exemplu, <verb> poate fi definit dupa cum urmeaza:

```
<verb>::="maninca"!"doarme"!"munceste"!etc.
```

unde bara verticala (!) inseamna "sau". In exact aceeasi maniera putem defini o instructiune BASIC:

```
<instructiune>::="LET"<variabila>=<expresie>
               !"GOTO"<numar de linie>
               !"PRINT"<lista de expersii> s.a.m.d.
```

Folositi extensiile de BASIC prevazute. Acestea duc la un cod mult mai rapid decit echivalentul lor BASIC.
Incercati sa nu utilizati acelasi tablou de mai multe ori si FOLOSITI CONSTANTE pentru a defini dimensiunile tabloului.
Incercati sa folositi variabile dintr-o singura litera orideciteori este posibil, deoarece BLAST-ul trateaza aceste variabile in mod special.



19. TOOLKIT-UL BLAST-ULUI

BLAST-ul este livrat impreuna cu un toolkit conceput sa ajute la dezvoltarea programelor.
TOOLKIT-ul se gaseste pe fata a doua a casetei. Pentru a-l incarca, tastati:

LOAD "TOOLKIT" <ENTER>

TOOLKIT-ul va porni automat si va semna cu mesajul:

BLAST TOOLKIT (C) OCSS 1985

La fel ca si compilatorul, TOOLKIT-ul se incarca in partea superioara a RAM-ului si pozitioneaza RAMTOP-ul sub zona pe care o ocupa. TOOLKIT-ul reduce memoria disponibila cu aproximativ 2K.
Nota: TOOLKIT-ul nu poate sa coexiste in RAM cu compilatorul.

Facilitatile disponibile sint listate mai jos. Fiecare functie este executata introducind un asterisc (*) urmat de o comanda formata dintr-o singura litera si un numar de parametrii.
In continuare n, n1 si n2 sint intregi.
Portiunea de program asupra careia va avea efect o anumita comanda e specificata de un interval de numere de linie dupa cum urmeaza:

n1-n2 inseamna liniile de la n1 la n2 inclusiv
n1- inseamna de la linia n1 la sfirsitul programului
-n2 inseamna de la inceputul programului la linia n2 inclusiv

Daca un interval de numere de linii e omis, toolkit-ul va considera ca acest interval cuprinde tot programul
Un punct (.) poate fi utilizat pentru a indica linia curenta.

20. COMENZI PENTRU LINII

1) EDIT sintaxa: *En1

Linia n1 e afisata pentru editare.

2) COPY sintaxa: *Cn1,n2

Copiaza linia n1 peste linia n2 care se pierde.

3) DELETE sintaxa: *Dn1

Sterge linia n1.

4) MOVE sintaxa: *Mn1,n2

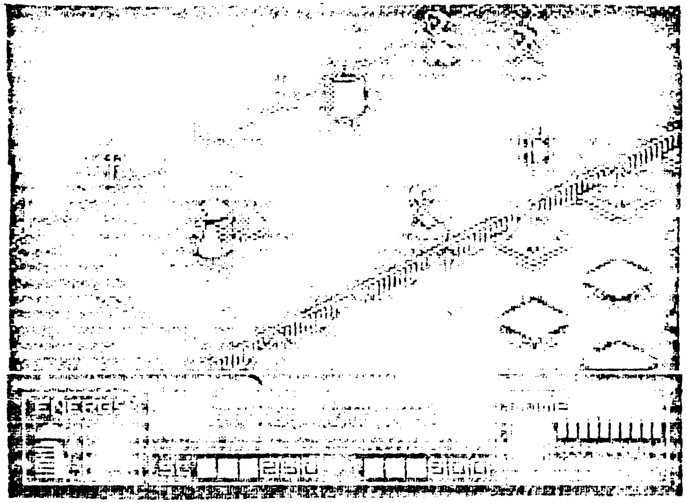
Muta linia n1 la linia n2, stergind linia n1.

21. COMENZI PENTRU BLOCURI

1) COPY sintaxa: *C<interval>,n

Copiaza liniile din intervalul specificat la linia n, scriind peste orice linie existenta. Liniile vor fi numerotate consecutiv, incepind de la linia n.

2) DELETE sintaxa: *D<interval>



Bineinteles, toti "non terminalii" trbuie definiti iar definirea completa a BASIC-ului de Spectrum e un fisier de sute de linii.

Pina acum am vazut cu se poate scrie SINTAXA unui limbaj. Pentru a face insa un compilator, trebuie sa gasim o cale pentru a exprima SEMANTICA, adica intelesul instructiunilor si actiunile pe care trebuie sa le ia computerul atunci cind recunoaste o instructiune. Notatia folosita nu o vom explica deoarece e mai complicata decit definirea sintaxei. Totusi ceea ce se intimpla in esenta e ca specificatiile referitoare la sintaxa impreuna cu cele privitoare la semantica sint oferite unui program numit METAPOD care le foloseste pentru a scrie codul compilatorului. METAPOD-ul, dezvoltat de OCSS, ruleaza pe un sistem UNIX multi-user si genereaza fisiere sursa scrise in limbajul C. Bineinteles, METAPOD-ul insusi este un fel de compilator; accepta un fisier sursa (definirea limbajului) si produce un fisier obiect (compilatorul), astfel putind fi utilizat pentru a se genera pe sine insusi. Asa a fost creat METAPOD-ul.

De ce folosim un generator de compilatoare ?

In special deoarece e mai ieftin. Scrierea "de mina" a unui compilator ia mult timp, iar produsul final e mult mai susceptibil sa contina erori. Un compilator generat, reflecta exact specificatiile cu care a fost alimentat generatorul de compilatoare si nu apare tentatia de a "taia coltul" in scrierea codului. Bineinteles, exista parti dintr-un compilator cum e BLAST-ul care trebuie scrise "de mina"; RTS-ul BLAST-ului a fost scris de mina pentru a creste viteza de rulare a programului. Intr-un compilator precum BLAST-ul aproximativ 70% din cod e generat automat.

Putem genera compilatoare pentru orice limbaj ?

Pentru majoritatea limbajelor, raspunsul e da. Bineinteles ca limbaje ca PASCAL, MODULA, C si BASIC fac parte din planurile noastre de viitor, desi putem produce si compilatoare dedicate pentru aplicatii cum ar fi controlul masinilor si robotilor, rapid si competitiv.

ANEXA 1

26. HARTA MEMORIEI PENTRU BLAST

```
PRAMT -----------------------------------------------------
    USER DEFINED GRAPHICS (grafice definite de utilizator)
UDG -------------------------------------------------------
                          BLAST
RAMTOP ----------------------------------------------------
                GOSUB STACK (stiva de GOSUB)
                MACHINE STACK (stiva masinii)
                  SPARE RAM (RAM liber)
STKEND ----------------------------------------------------
           CALCULATOR STACK (stiva calculatorului)
STKBOT-----------------------------------------------------
              WORKSPACE (spatiu de lucru)
WORKSP ----------------------------------------------------
             EDITING AREA (zona de editare)
E-LINE ----------------------------------------------------
          BASIC VARIABLES (variabile BASIC)
VARS ------------------------------------------------------
             BASIC PROGRAM (program BASIC)
PROG ------------------------------------------------------
       CHANNEL INFORMATION (informatii de canal)
CHANS -----------------------------------------------------
       MICRODRIVE MAPS (harti pentru microdrive)
 INTERFACE 1 SYSTEM VARIABLES (var. de sist. pt. interf. 1)
          SYSTEM VARIABLES (variabile de sistem)
          PRINTER BUFFER (tampon pt. imprimanta)
          ATTRIBUTES (atribute de culoare)
            DISPLAY FILE (memorie video)
                         ROM
-----------------------------------------------------------
```

27. HARTA MEMORIEI LA RULARE

```
PRAMT -----------------------------------------------------------
      USER DEFINED GRAPHICS (grafice definite de utilizator)
UDG, RAMTOP -----------------------------------------------------
                 GOSUB STACK (stiva de GOSUB)
                 MACHINE STACK (stiva masinii)
                   SPARE RAM (RAM liber)
STKEND ----------------------------------------------------------
            CALCULATOR STACK (stiva calculatorului)
STKBOT ----------------------------------------------------------
                 WORKSPACE (spatiu de lucru)
WORKSP ----------------------------------------------------------
                EDITING AREA (zona de editare)
E-LINE ----------------------------------------------------------
           RUN TIME VARIABLES (variabile de rulare)
VARS ------------------------------------------------------------
              BLASTED PROGRAM (program BLAST-at)
PROG ------------------------------------------------------------
           CHANNEL INFORMATION (informatii de canal)
CHANS -----------------------------------------------------------
             MICRODRIVE MAPS (harti pt. microdrive)
         RUN TIME SYSTEM (sistem de rulare (RTS-ul))
-----------------------------------------------------------------
   INTERFACE 1 SYSTEM VARIABLES (var. de sist. pt. interf. 1)
            SYSTEM VARIABLES (variabile de sistem)
            PRINTER BUFFER (tampon pt. imprimanta)
              ATTRIBUTES (atribute de culoare)
                DISPLAY FILE (memorie video)
                            ROM
-----------------------------------------------------------------
```

ERATA

N.T. In documentatia originala referirile din erata s-au facut la diverse numere de pagina. Deoarece documentatia de fata s-a redactat pe un editor ce nu suporta numerotarea automata a paginilor, sarcina punerii in corespondenta a celor de mai jos cu un anumit capitol, revine in mod ingrat cititorului.

Cantitatea de memorie disponibila pentru codul sursa este de 2K si nu 5K. Bineinteles, un program de orice marime poate fi compilat de pe banda pe banda sau de pe microdrive pe microdrive.

Retineti ca directiva *N sterge din memorie atit programul sursa cit si cel obiect.

Comanda *S poate fi folosita doar pentru a salva un program compilat pe caseta. Utilizatorii posesori de microdrive vor gasi ca e mai usor sa se compileze de pe microdrive pe microdrive.

Cind BLAST-ul compileaza pe microdrive se scriu doua sau trei fisiere, necesare pentru rularea programului. Numele lor este:

- numele de fisier al codului obiect (cel specificat).
- acelasi nume cu .P anexat
- un fisier optional .V (vezi capitolul de variabile salvate la sfirsitul manualului).

Programul compilat se executa incarcind primul dintre acestea (cu numele specificat de utilizator) si tastind RUN.

Capitolul referitor la copierea programelor BLAST-ate este incorect. In schimb, s-a furnizat un program pentru copierea programelor BLAST-ate pe fata a doua a casetei, imediat dupa toolkit. Pentru a incarca acest program tastati:

    LOAD "COPIER"

si urmati instructiunile programului.

Retineti ca toate comenzile (*C, *N, etc.) pot fi tastate atit cu majuscule cit si cu minuscule.

Comanda *B (BACKUP) nu a fost implementata. In schimb s-a prevazut optiunea de copiere a BLAST-ului pe microdrive imediat dupa incarcarea programului.

Datorita penuriei de memorie, extensiile de BASIC nu au fost implementate. Drept compensatie, toolkit-ul a fost prevazut cu cinci noi comenzi (vezi capitolul de informatii aditionale la sfirsitul manualului).

TOOLKIT

Comanda:

    RANDOMIZE USR 60500

va reactiva toolkit ul dupa comanda *Q.

Comanda *D fara parametrii va sterge intreg programul lasind variabilele intacte.

In comenzile *F si *S este imposibila introducerea directa a cuvintelor cheie BASIC. Exista totusi un artificiu pentru rezolvarea problemei. Cind e nevoie de un cuvint cheie, se tasteaza intii THEN si apoi cuvintul cheie. Apoi se muta cursorul inapoi si se sterge THEN.

In comenzile SEARCH SI REPLACE:

i. Separatorul trbuie sa fie o virgula.
ii. Sirul care e inlocuit nu poate fi vid.
iii. Liniile modificate nu sint verificate sintactic.

Daca intervalul de numere de linii e omis dintr-o comanda a toolkit-ului,rebuie inclusa orice virgula care ar urma intervalului respectiv.

Comanda *B (BLAST SAVE) se foloseste cu caseta. BLAST-ul poate compila un program care a fost salvat pe microdrive in mod normal.

Comanda TRACE (*T) (de urmarire) nu are ca parametru un numar de linie. *T va activa optiunea TRACE care va ramine operativa pina la dezactivarea ei cu comanda *U.

Functia de renumerotare nu va functiona la mai mult de 643 de linii odata. Programele lungi pot fi renumerotate in doua sau mai multe etape.

PROBLEME

Foarte rar, BLAST-ul poate obiecta unei linii de BASIC care apare corecta la inspectare. Daca se intimpla asa ceva, BLAST-ul se va opri cu mesajul:

WARNING - HIT ANY KEY

Deindata ce s-a apasat o tasta, BLAST-ul va face ca acea instructiune sa fie trecuta interpretorului la rulare si va continua compilarea. Desi acest eveniment va fi etichetat de BLAST ca un avertisment, el nu va afecta rularea programului final. De asemenea, BLAST-ul poate fi fortat sa treaca o linie interpretorului inserindu-i REM% la inceput (vezi manualul). Acest lucru poate fi util daca o anumita linie da probleme la compilare. Aceasta facilitate nu poate fi folosita pentru functiile definite de utilizator.

EXTENSII DE BASIC

Din pacate, din cauza lipsei de memorie, acestea nu au fost implementate. In schimb, ca o compensatie, s-a extins toolkit-ul cu inca cinci comenzi.

*V listeaza diferite variabile de sistem utile, inclusiv cantitatea de memorie ramasa libera.

*L listeaza toate variabilele BASIC, definite in mod curent, impreuna cu valoarea lor.

(*J <nr. de linie> se alatura liniei indicate ulterior.

*G si *A. In mod normal operatiile de cautare si gasire se vor opri si vor astepta ca utilizatorul sa apese o tasta dupa fiecare gasire sau substitutie. Pentru a dezactiva aceasta modalitate, folositi *G iar pentru a reveni la vechiul mod, folositi *A.

VARIABILE SALVATE (DOAR PENTRU MICRODRIVE)

Pentru a economisi spatiu, multe programe BASIC sint salvate impreuna cu o parte din variabile. BLAST-ul trateaza o astfel de eventualitate folosind directiva REM! AUTORUN dupa cum urmeaza:

Daca directiva AUTORUN apare la inceputul programului, BLAST-ul va crea un fisier separat (numit .V) care contine variabile salvate. Cind se incarca programul BLAST-at, acest fisier va fi adus in memorie automat. Daca directiva AUTORUN nu este inclusa, BLAST-ul va considera ca nu are de salvat variabile.

O alta metoda, aplicabila si pentru banda, si pentru microdrive, e urmatoarea:

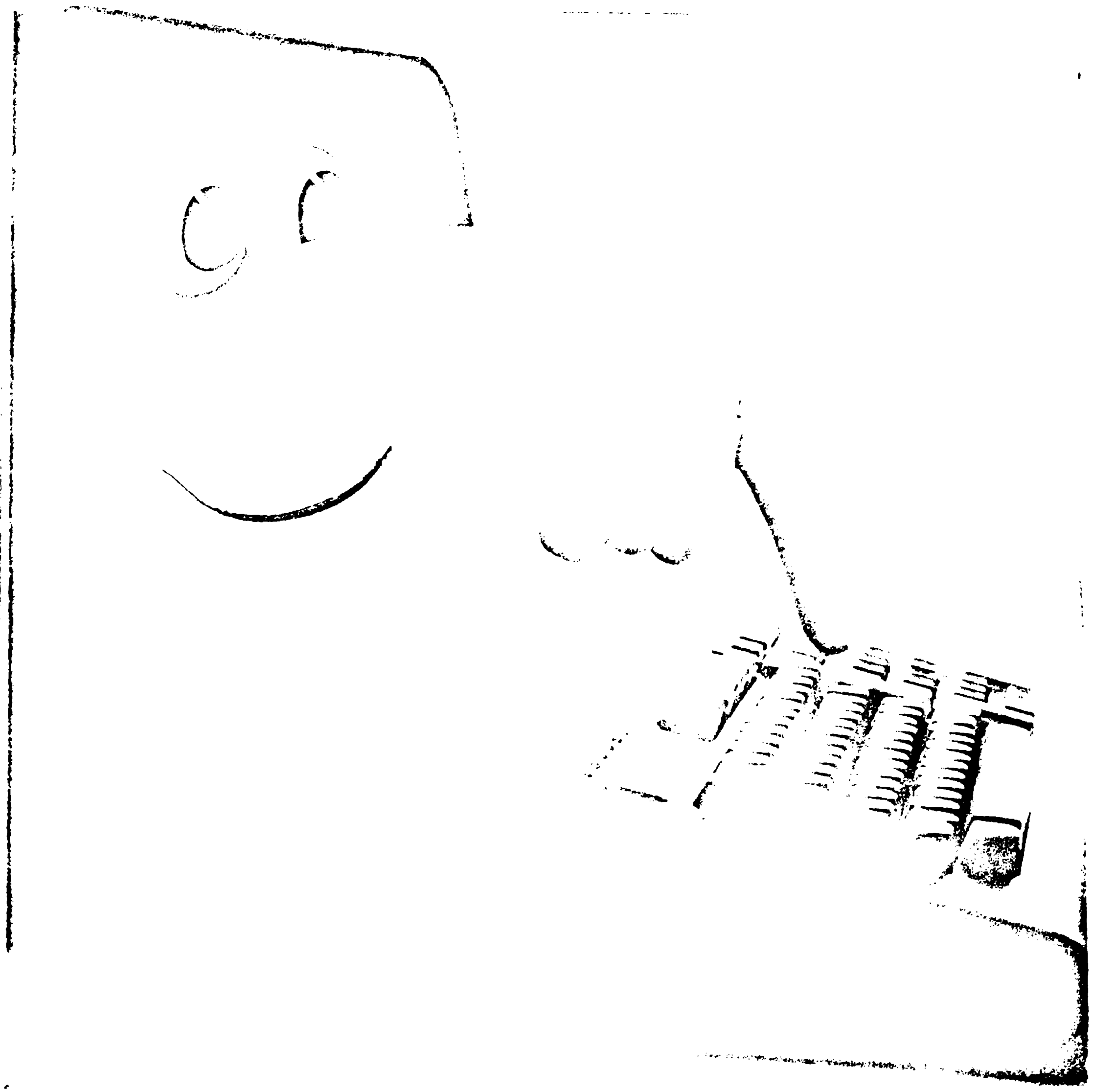
Se incarca toolkit-ul si programul BASIC de compilat. Se foloseste comanda *D pentru a sterge programul si apoi se salveaza variabilele sub un nume adecvat, ca un program normal. Se insereaza o linie la inceputul programului de compilat pentru a face MERGE cu aceste variabile (N.T. vazute ca un program BASIC). Cind programul BLAST-at va rula, el va face merge cu aceste variabile.

ATENTIUNE !!!
NU PUTETI FOLOSI BLAST-UL FARA ACEASTA TABELA DE DECODIFICARE
NU O PIERDETI !!!

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	
40	R	Y	G	R	Y	G	G	R	Y	G	W	G	G	Y	R	R	Y	G	Y	G	Y	G	W	Y	G	G	40
39	Y	G	R	Y	W	G	Y	G	G	R	G	Y	R	G	Y	G	G	R	G	R	G	Y	G	R	G	R	39
38	G	Y	G	G	Y	R	R	G	Y	G	Y	R	G	Y	W	Y	R	Y	R	G	R	G	R	G	Y	G	38
37	R	G	Y	G	R	Y	G	G	R	Y	W	Y	R	G	Y	G	W	G	Y	G	Y	G	Y	G	G	R	37
36	Y	R	G	G	Y	G	R	Y	G	G	G	R	G	Y	G	R	G	R	G	W	G	Y	G	R	G	Y	36
35	G	G	Y	Y	G	W	Y	W	Y	G	R	Y	R	G	R	G	Y	G	Y	R	Y	G	W	Y	R	Y	35
34	R	Y	G	R	R	Y	G	Y	R	Y	G	Y	Y	G	Y	Y	G	Y	R	G	G	R	Y	G	Y	G	34
33	Y	G	R	Y	G	R	Y	G	G	R	G	G	G	R	G	R	Y	G	Y	G	Y	G	W	Y	R	G	33
32	R	G	Y	G	Y	G	R	Y	R	Y	W	G	G	Y	R	G	W	Y	G	Y	R	G	Y	R	Y	G	32
31	G	Y	W	G	R	Y	G	R	G	G	R	Y	G	R	Y	R	G	Y	G	R	Y	R	G	G	R	G	31

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	
30	Y	R	G	Y	G	R	Y	R	Y	R	G	G	R	Y	G	Y	R	Y	R	Y	W	Y	G	Y	G	W	30
29	R	Y	R	G	W	G	G	G	Y	R	Y	W	Y	G	R	G	G	R	Y	R	G	G	W	G	R	Y	29
28	G	R	G	Y	G	G	Y	R	G	Y	G	Y	G	Y	G	R	R	Y	G	Y	R	R	Y	R	G	G	28
27	R	Y	G	R	Y	R	R	Y	R	G	G	G	W	G	R	Y	G	R	W	G	G	Y	R	G	R	Y	27
26	Y	R	G	G	Y	G	Y	R	G	Y	R	Y	G	R	Y	R	G	R	G	G	Y	W	Y	G	Y	R	26
25	G	Y	R	Y	R	W	G	Y	R	G	G	R	Y	R	G	Y	G	G	Y	R	G	Y	G	R	G	Y	25
24	R	G	G	R	G	Y	G	R	Y	W	G	R	G	G	R	G	W	Y	G	G	R	G	Y	G	Y	R	24
23	G	G	G	Y	R	G	R	Y	G	G	Y	Y	R	Y	G	R	G	G	W	Y	G	R	R	Y	G	Y	23
22	Y	W	Y	R	Y	G	Y	G	G	Y	G	R	Y	R	G	Y	R	Y	G	R	R	G	Y	W	Y	G	22
21	G	G	R	G	R	Y	G	Y	W	G	R	Y	G	Y	G	G	Y	R	R	Y	G	R	R	G	R	Y	21
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	
20	R	G	Y	Y	G	Y	R	Y	G	R	Y	R	Y	W	Y	R	R	G	Y	G	R	Y	G	W	Y	R	20
19	Y	G	R	G	G	W	Y	G	Y	G	R	G	W	G	W	G	G	W	G	W	G	G	Y	R	R	G	19
18	R	Y	G	R	G	Y	R	G	R	R	Y	W	Y	G	Y	R	G	R	Y	G	Y	G	R	G	R	Y	18
17	Y	W	G	Y	R	G	R	G	Y	G	G	Y	G	W	G	Y	R	Y	G	Y	G	Y	G	G	Y	R	17
16	G	G	G	R	G	R	Y	R	G	R	G	R	G	Y	G	G	G	G	G	G	R	G	R	Y	R	G	16
15	Y	R	G	Y	R	Y	W	Y	G	R	R	Y	R	R	Y	R	G	R	Y	R	Y	R	G	W	Y	G	15
14	G	Y	R	R	G	G	Y	G	G	R	G	W	G	W	R	G	Y	G	W	G	Y	G	Y	G	G	Y	14
13	G	G	Y	G	Y	G	Y	W	Y	G	Y	G	W	Y	G	R	G	Y	G	Y	G	Y	G	Y	R	G	13
12	R	Y	R	G	G	R	G	G	R	Y	R	G	W	G	W	Y	G	R	G	G	Y	R	G	W	G	Y	12
11	Y	R	Y	W	G	Y	G	R	Y	W	Y	G	G	Y	G	R	Y	G	Y	G	R	Y	R	Y	G	R	11
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	
10	R	G	G	Y	Y	G	R	G	R	Y	G	R	Y	W	Y	G	K	G	G	W	G	W	G	K	G	Y	10
9	G	Y	G	R	G	R	Y	G	G	G	Y	R	G	Y	G	G	Y	R	Y	G	W	Y	G	Y	Y	G	9
8	R	G	W	Y	R	Y	G	R	Y	R	R	G	R	G	R	Y	G	Y	G	G	Y	G	R	G	R	W	8
7	Y	R	G	G	Y	W	G	Y	G	G	Y	G	Y	R	Y	Y	W	G	Y	R	G	R	Y	R	Y	G	7
6	G	Y	R	Y	R	G	G	G	R	G	G	R	R	G	R	G	G	Y	R	Y	W	Y	G	Y	R	G	6
5	Y	W	G	G	G	Y	Y	W	G	Y	W	G	G	R	G	Y	R	G	Y	G	Y	G	Y	G	Y	R	5
4	G	G	Y	R	G	R	G	G	Y	R	G	Y	R	G	Y	W	Y	G	R	Y	R	G	W	Y	R	G	4
3	R	Y	R	G	Y	G	R	Y	R	Y	G	R	G	Y	G	G	R	R	G	R	G	Y	G	R	G	Y	3
2	Y	G	G	G	R	G	Y	G	G	R	Y	G	R	G	R	Y	R	Y	R	Y	R	G	Y	R	Y	G	2
1	W	Y	R	Y	G	Y	W	Y	G	Y	G	R	Y	R	G	W	Y	G	Y	G	G	R	Y	G	G	W	1
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	

N.T. Am incercat ca traducera acestei documentatii sa fie lipsita de erori de orice natura. Cu toate acestea ele ar putea sa existe, in special in tabelul final. Pentru eventualul RESET al sistemului in cazul introducerii gresite a codurilor de protectie va cer scuze anticipat.



MANEA TITUS

KSEROKS V3.0

Kseroks V3.0 este primul program de copiere din ROM comparabil cu cele din YU si UK.

Optiuni de lucru:

LOAD-Permite incarcarea fisierelor cu 'header'.
DATA-Incarca fisiere fara 'header'.
EXIT-Reintoarcere in BASIC.
FLEN-Incarca fisiere cu 'flag' cunoscut si lungime cunoscuta.
SAVE-Salveaza fisierele incepind cu primul.
TEST-Verifica fisierele salvate.Cele eronate vor fi semnalate prin aparitia unui blinker la capat de rind.
VIEW-Optiuni relative la un singur fisier.
RESET-Sterge toate fisierele.
HEADER-Citeste doar informatia din 'header'.
MAXBYT-Incarca fisiere cu lungime <=49075.Optiuni in MAXBYT : D-DATA.S-SAVE,A+G-EXIT.

Optiunile comenzii VIEW:

S-SAVE,T-TEST,K-DELETE.V-Trece la urmatorul fisier.
F-Se introduce/schimba 'flag'-ul.
N-Se introduce/schimba numele.
A-Listare in ASCII.
L-Listare BASIC-numai pentru programe BASIC.
R-AUTO RUN OFF-numai pentru programe BASIC.
3-Pune/sterge pauza la inceputul salvarii.Pauza e marcata prin aparitia unui diez '#'.

Optiuni in listare:

ENTER-listare rapoda,P-listarea unei singure linii,SPACE-revenire in meniu.

Fisierele incarcate cu eroare se pot pastra prin apasarea tastei'I'.

ATENTIUNE!! Celelalte versiuni ale programului sint eronate!

TIBERIU ONU

LASER GENIUS

Nota : Aceste insemnari despre pachetul LASER GENIUS al firmei OASIS reprezinta traducerea unui articol aparut in numarul 24 al revistei RACUNARI sub semnatura lui Zeljko Juric. Ele nu au pretentia unei documentatii complete si se refera doar la asamblorul LASER,pseudocompilatorul PHOENIX si pachetul de rutine ajutatoare TOOLKIT. Pachetul LASER GENIUS mai cuprinde in afara de acestea si un dezasamblor/monitor cu performante deosebite (in 2 versiuni),precum si un analizor RPN. Traducerea din limba sirbocroata si editarea au fost realizate de catre Tiberiu Onu,care va cere anticipat scuze pentru eventualele greseli strecurate in text.

V-ati gindit vreodata ca popularul GENS nu merita o nota de trecere,in primul rind din cauza editorului care ingreuneaza substantial munca programatorului ? De asemenea,v-ati gindit ca binecunoscutul debugger MONS poate fi folosit mai bine la vinatoarea de ursi decit la cea de bug-uri ? Daca raspunsul la aceste doua intrebari a fost pozitiv,rezolvarea este foarte simpla : luati pachetul DEVPACK 3 , aruncati-l la lada cu vechituri si procurati LASER GENIUS - pachet de programe pentru lucrul in cod masina. Acest pachet cuprinde un asamblor/editor,pseudocompilatorul PHOENIX,pachetul de rutine ajutatoare TOOLKIT,doua versiuni de dezasamblor/monitor,analizorul RPN si alte citeva programe utilitare. LASER GENIUS este opera programatorilor firmei OASIS Software care au intrat deja in legenda datorita programelor MACHINE LIGHTNING,WHITE LIGHTNING si LASER BASIC.

Pe piata exista trei versiuni ale pachetului LASER GENIUS,care se deosebesc in anumite privinte. Prima versiune a fost scoasa pentru calculatoarele AMSTRAD/SCHNEIDER CPC 464,664 si 6128 , cea de-a doua pentru SPECTRUM 48 si SPECTRUM+ ,iar cea de-a treia pentru calculatoarele SPECTRUM 128 si SPECTRUM +2. Diferentele intre versiuni vor fi mentionate in text. In afara de cele mentionate ,autorul este convins ca programul MAGUS ,evidentiat in toate cataloagele de programe pentru SPECTRUM 128 este unul si acelasi cu aceasta a treia versiune a pachetului LASER.

PUNERE IN FUNCTIUNE

Dupa ce introducem comanda LOAD ,se incarca un loader BASIC (aproximativ 3K) care ne intreaba daca dorim o copie a programului pe microdrive (in cele ce urmeaza,referirile la microdrive vor fi legate de calculatoare SPECTRUM iar referirile la disc se vor face pentru calculatoare AMSTRAD) ,daca dorim modificarea culorilor ecranului si a tipului de imprimanta utilizat si daca dorim ca impreuna cu asamblorul sa se incarce si pseudocompilatorul PHOENIX sau pachetul TOOLKIT (functionarea acestora este conditionata de existenta asamblorului in memorie).

Dupa aceste intrebari plictisitoare asteptam incarcarea ,care dureaza citeva secunde daca dispunem de microdrive sau discdrive sau mai mult decit un minut daca ne folosim de casetofon.

Unul din neajunsurile mari ale programului este consumul im-

portant de memorie (acest neajuns este nesuparator insa in cazul in care lucram cu AMSTRAD 6128 sau SPECTRUM 128). Asamblorul singur ocupa 23 Ko de memorie. Daca mai incarcam si pachetul TOOLKIT (1,5 Ko) si pseudocompilatorul PHOENIX (3 Ko) (ambele insa pot lipsi) si daca luam in considerare si diferitele spatii de lucru si buffere folosite de program ,observam ca ne ramine foarte putin loc pentru fisierul text si pentru codul obiect. Din fericire insa ,programul a fost astfel conceput incit fisierul sursa este construit intr-o forma incredibil de compacta (de 6-7 ori mai scurt decit un fisier GENS echivalent) iar comenzile INCLUDE si OPENOUT (mai cu seama pentru microdrive si disc) sint excelent impementate.

In total ,versiunea pentru SPECTRUM 128/+2 ocupa 32 Ko de memorie ,deoarece TOOLKIT si PHOENIX se incarca intotdeauna impreuna cu asamblorul si aceasta versiune poseda comenzi suplimentare. Versiunea pentru 128 Ko se incarca in RAM-disc si lucreaza prin paginari ,astfel ca nu consuma nimic din memoria 'normala'. Fisierul sursa este de asemenea stocat in RAM-disc.

EDITORUL

Dupa incarcare ,loader-ul BASIC se sterge si RAMTOP-ul se pozitioneaza la adresa 25500 (la SPECTRUM).

Inca de la inceput ne gasim in fata unui minunat SCREEN - editor ,care lucreaza pe 42 (la AMSTRAD 40) de coloane si care utilizeaza un buffer special de stocare a textului inserat. Initial acest buffer are o dimensiune de 10 Ko ,dar cu ajutorul unor comenzi speciale marimea sa poate fi redusa la doar 1Ko ,pentru a asigura un spatiu de memorie suplimentar pentru codul obiect si fisierul sursa. Bufferul editorului foloseste la stocarea oricaror comenzi introduse si nu trebuie confundat cu spatiul de lucru necesar construirii fisierului sursa. Ecranul se prezinta ca si o fereastra interioara acestui buffer ,textul care a iesit din ecran putind fi revazut folosind cursorul.

Foarte interesanta este capacitatea editorului de a permite introducerea de linii care contin mai multe instructiuni in limbaj de asamblare. Daca intoducem de exemplu linia :

```
10 loop: DEC BC (ENTER) LD A,B (ENTER) OR C (ENTER) JR NZ,loop
(ENTER)
```

si apoi :

```
20 RET (ENTER)
```

listingul va avea urmatorul aspect :

```
10 loop : DEC BC
          LD  A,B
          OR  C
          JR  NZ,loop
20        RET
```

Din acest motiv ,comanda AUTO nu este necesara. O linie poate avea o lungime de pina la 10 ecrane.

De asemenea ,nu exista nici tasta pentru tabulare ,deoarece tabularea se executa automat.

Lungimea maxima a unei etichete este de 240 de caractere. Numele unei etichete poate contine si caracterele '.' sau '$' si se sfirseste cu caracterul ':' care slujeste doar ca separator si nu face parte din nume.

Comentariile sint in continuare precedate de caracterul ';'

La editare avem la dispozitie urmatoarele taste de control (cele din paranteza sint valabile pentru AMSTRAD) :

CURSOR	deplasarea cursorului
CAPS 4 (SHIFT 9)	cursorul la anterioara pozitie tabulata
CAPS 9 (TAB sau SHIFT RIGHT)	cursorul la urmatoarea pozitie tabulata
CAPS 0 (FEL)	sterge caracterul din stinga cursorului
SYM 0 (CLR)	sterge caracterul din dreapta cursorului
CAPS 3 (SHIFT DEL)	sterge rindul care contine cursorul
SYM F (CTRL DEL)	sterge de la cursor la capatul rindului
SYM I (SHIFT TAB)	introduce un rind gol
SYM W (COPY)	introduce un spatiu
CAPS 1 (CTRL TAB)	porneste/opreste modul INSERT
SYM G (CTRL 'L')	sterge ecranul si bufferul de editare

La AMSTRAD exista si o schimbare specifica a cursorului care se obtine cu ajutorul tastelor SHIFT si CTRL. La SPECTRUM sint interesante combinatiile SYM Q si SYM E. Comanda SYM A foloseste la oprirea oricarei operatii.

Dupa introducerea fiecarei instructiuni ,se realizeaza verificarea sintactica si se afiseaza comentariile de rigoare. De exemplu ,introducind LD H,(BC) vom primi mesajul 'illegal second operand' si cursorul va fi pozitionat in locul in care a fost perceputa eroarea. Dupa corectarea erorii ,mesajul va disparea de pe ecran. In fiecare situatie asamblorul va raporta foarte detaliat toate erorile de sintaxa intervenite.

PARTICULARITATILE ASAMBLORULUI

GENASM este un macroasamblor in doua treceri ,realizat dupa toate standardele ZILOG. Parametrii numerici ai instructiunilor Z 80 pot fi specificati ca si constante sau ca si expresii.

Constantele pot fi : - zecimale (ex: 239)
- binare (ex: %101101)
- octale
- hexa (ex: #5C3A sau 5C3AH)
- alfanumerice (ex: "A")

Expresiile pot contine constante ,etichete ,numaratorul de locatii ($) si numaratorul de adrese (.) (acestea din urma vor fi explicate ulterior). In cadrul expresiilor se permite utilizarea urmatorilor operatori (cifrele din paranteza reprezinta prioritatea) :

OPERATORI BINARI

+	(adunare , 7)
-	(scadere , 7)
*	(inmultire , 8)
/	(impartire intreaga , 8)
%	(restul impartirii modulo ,8)
>	(mai mare decit , 5)
<	(mai mic decit , 5)
>=	(mai mare sau egal , 5)
<=	(mai mic sau egal , 5)
?=	(egal , 4)
!=	(inegal , 4)
<<	(shift-are spre stinga , 6)
>>	(shift-are spre dreapta , 6)
@<	(rotatie spre stinga , 6)
@>	(rotatie spre dreapta , 6)
&	(AND binar , 3)
!	(OR binar , 3)
^	(XOR binar , 3)
&&	(AND logic , 2)
!!	(OR logic , 2)

OPERATORI UNARI

-	(minus unar , 9)
!	(NOT logic , 9)
^	(NOT-CPL binar , 9)
*	(continutul adresei-pe 16 biti , 9)

Un bun cunoscator al limbajelor de programare va recunoaste multi operatori imprumutati din limbajul C.

Se observa ca LASER ,spre deosebire de GENS ,are prioritati ale operatiilor (9 este prioritatea cea mai mare). Pot fi folosite si paranteze,dar cu grija spre a nu fi confundate cu parantezele limbajului de asamblare Z 80.

Numaratorul de adrese (.) are valoarea adresei la care se stocheaza momentan codul obiect ,iar numaratorul de locatii ($) contine valoarea curenta a contorului de locatii ,conform directivei ORG. Aceste doua numaratoare au de obicei (dar nu totdeauna) aceeasi valoare.

In cadrul evaluarii expresiilor ,in timpul asamblarii ,fiecare eroare intervenita se raporteaza textual (de ex: 'number out of range -128,127' si nu doar ERROR 10 ca si la GENS). In total exista 32 de tipuri erori de sintaxa si 30 de tipuri de erori ce pot aparea in timpul rularii ,clasificate dupa importanta in 3 categorii : WARNING (avertisment) ,ERROR (eroare) si FATAL ERROR (eroare fatala)

MACROINSTRUCTIUNI

Macroinstructiunile reprezinta o arma forte a asamblorului GENASM. Macroinstructiunile LASER nu sint atit de puternice si flexibile ca si cele din MACHINE LIGHTNING ,dar sint deopotriva de utile.

Definirea macroinstructiunilor se face prin comanda MACRO intr-o sintaxa de forma :

eticheta : MACRO \ parametri

Caracterul '\' (backslash) foloseste ca si separator care specifica inceperea parametrilor macroinstructiunii.

Iata in continuare doua exemple care vor demonstra modul de utilizare a macroinstructiunilor :

outchar : MACRO \ code

```
        LD   A,code
        RST  16
        ENDM
```

Macroinstructiunea de mai sus va tipari un caracter dat.
Urmatoarea macroinstructiune schimba intre ele valorile a doua registre pereche (intr-un mod destul de neelegant) :

```
swap : MACRO \ reg1,reg2
       PUSH   reg1
       PUSH   reg2
       POP    reg1
       POP    reg2
       ENDM
```

Macroinstructiunile astfel definite pot fi ulterior utilizate in cadrul unui program in cod masina ,fiind apelate astfel:

```
outchar 65     sau
outchar "A"    sau
outchar "A"+1  sau
swap HL,BC
```

Evident ,ca si parametri ai macroinstructiunilor pot fi folosite si expresii ,dar constructii de tipul :

```
LD A,code+1
```

nu sint permise in definirea macroinstructiunilor.

COMENZI DIRECTE ALE EDITORULUI

In continuare se prezinta comenzile directe pe care le accepta editorul GENASM. In cadrul celor de mai jos ,x$ y$ ... reprezinta stringuri arbitrare ,n n1 n2 ... reprezinta constante zecimale ,x x1 x2 ... reprezinta expresii arbitrare ,iar b b1 b2 ... reprezinta blocuri de program definite in stil COMMODORE (de ex : 10-50 sau -50 sau 10- sau 30 samd). Blocul implicit de program este cel dintre liniile 0-65534.

CLS	sterge ecranul si bufferul de editare
SETSPACE n	stabileste dimensiunea spatiului de lucru al editorului ,in octeti. La SPECTRUM 128 aceasta comanda nu este prevazuta din cauza spatiului suficient de memorie.
LIST [b]	listeaza programul sau o parte a sa.
LLIST [b]	ca si LIST ,dar la imprimanta.
DELETE [b]	sterge un bloc de linii.
COPY b1,b2	copiaza blocul de linii b1 la pozitia b2.
MOVE b1,b2	muta blocul de linii b1 la pozitia b2,
RENUM [n1[,n2[,n3]]]	renumeroteaza programul astfel incit prima linie sa fie n1 ,iar pasul n2. Renumerotarea se executa incepind de la linia n3. (Implicit : n1=n2=10 , n3=0).
FIND x$ [,b]	cauta stringul x$ in blocul de linii b.
REPLACE x$,y$ [,b]	inlocuieste stringul x$ cu y$ in blocul de linii b. Inlocuirea se executa optional la afisarea mesajului : 'replace (y/n) ?' .
NEXT	continua executia comenzilor FIND si REPLACE daca aceasta a fost intrerupta.
PRINT x [,n]	afiseaza valoarea expresiei x in baza n ,in notatie cu semn. (Implicit n=16).
UPRINT x [,n]	afiseaza valoarea expresiei x in baza n ,in notatie fara semn. (Implicit n=16).
BASE n	stabileste valoarea implicita a bazei pentru comenzile PRINT si UPRINT. Valori posibile ale lui n sint : 2,8,10 si 16.
FORM	trimite un caracter de 'form feed' la imprimanta.
WIDTH [n]	stabileste latimea paginii la imprimanta. (Implicit n=65536).

LENGTH [n]	stabileste numarul de rinduri pe pagina la imprimanta. (Implicit n=65536).
MARGIN [n]	stabileste pozitia marginii din stinga la imprimanta. (Implicit n=0).
SAVE [x$ [,b]]	salveaza blocul de linii b pe disc, casetofon sau microdrive. In cazul in care se utilizeaza casetofonul ,salvarea se face in blocuri a cite 2 Ko, ceea ce este necesar comenzii INCLUDE (Implicit x$=numele ultimului fisier incarcat).
LOAD x$ [,b]	incarca programul x$ si il plaseaza la pozitia b.
VERIFY x$ [,b]	verifica salvarea pe banda a programului sau a unei portiuni de program.
CODE x$,x1,x2	salveaza zona de memorie dintre adresele x1 si x2 ,sub numele x$. Versiunea pentru SPECTRUM 128 mai admite si un al patrulea parametru optional , care selecteaza pagina de RAM.
CAT	face catalogul programelor de pe disc sau cartridge. La AMSTRAD ,comanda functioneaza si pentru casetofon.
ERA x$	sterge articolul x$ de pe cartridge sau disc.
TAPE	selecteaza casetofonul ca si dispozitiv de intrare/iesire.
TAPE.IN	selecteaza casetofonul ca si dispozitiv de intrare.
TAPE.OUT	selecteaza casetofonul ca si dispozitiv de iesire.
MDRV [n]	selecteaza microdrive-ul numarul n ca si dispozitiv de intrare/iesire. (Implicit n=1). La AMSTRAD se utilizeaza comenzile DRIVEA si DRIVEB la selectarea unitatii de disc.
MDRV.IN	selecteaza microdrive-ul ca si dispozitiv de intrare. La AMSTRAD DISC.IN.
MDRV.OUT	selecteaza microdrive-ul ca si dispozitiv de iesire. La AMSTRAD DISC.OUT.
REN x$,y$	schimba numele programului din x$ in y$ pe disc (doar la AMSTRAD).
ASSEM	asambleaza programul.
ASSEMC	asambleaza programul fara stergerea valorilor etichetelor ,ramase de la asamblarea anterioara.
TABLE [n]	afiseaza tabela de simboluri (numele si valorile asociate etichetelor) in format ASCII. Parametrul n stabileste formatul de tiparire. (Implicit n=16)
LTABLE [n]	ca si TABLE ,dar la imprimanta.
TABLEN [n]	afiseaza tabela de simboluri in format numeric.
LTABLEN [n]	ca si TABLEN ,dar la imprimanta.
MISSING [n]	tipareste numele tuturor etichetelor care au ramas nedefinite in decursul asamblarii.
LMISSING [n]	ca si MISSING ,dar la imprimanta.
UNUSED [n]	tipareste numele si valorile tuturor etichetelor care au fost definite in program ,dar nu au fost niciunde folosite.
LUNUSED [n]	ca si UNUSED ,dar la imprimanta.
CLEAR	sterge tabela de simboluri.
EXECUTE x [,p1,p2,...]	Porneste programul masina de la adresa x. Parametrii optionali p1,p2,... pot fi de orice tip (de ex: EXECUTE start,"Video Byte Studios",5,%10110,"prog") si se adreseaza prin registrul IX. Registrul A contine numarul parametrilor. Versiunea pentru SPECTRUM 128 mai admite un parametru care se-

lecteaza pagina RAM in care rezida programul.

STATS — prezinta harta memoriei (unde se gaseste asamblorul,fisierul sursa,tabela de simboluri,bufferele,samd). La SPECTRUM 128 se prezinta si harta RAMDISC-ului.

EXIT — intoarcere in BASIC. Revenirea in asamblor se face cu RANDOMIZE USR 65533.

HOUSEWORK — apeleaza un program de COPY/SEARCH incorporat asamblorului ,care serveste la copierea fisierelor cu ajutorul lui LASER. Acest program dispune de 10 comenzi proprii ,dintre care cea mai importanta este COPY x$,y$. Intoarcerea in editor se face prin comanda EXIT.

COMENZILE PACHETULUI TOOLKIT

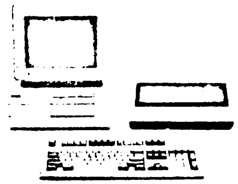
LOADASCII x$ [,n1 [,n2]] — incarca si recodifica fisierele care nu au fost scrise cu ajutorul lui LASER ,astfel incit sa apara intr-un format specific acestui program. x$ reprezinta numele fisierului ,n1 reprezinta optiunea de recodificare (implicit n1=0) ,iar n2-numarul de instructiuni reunite in cadrul unei linii (implicit n2=10).

In total exista 16 optiuni de recodificare (0-15) ,fiecare din cei patru biti ai optiunii specificind o functie speciala in recodificare.

La recodificarea fisierelor GENS se utilizeaza pentru SPECTRUM n1=12 ,iar pentru AMSTRAD n1=8.

Greselile de sintaxa rezultate in urma recodificarii ,vor fi specificate prin 'bad line'.

ASSEML — comanda foarte complicata ,care realizeaza asamblarea selectiva a unor rutine din biblioteca utilizatorului.



Explicatiile referitoare la efectul acestei instructiuni ocupa 2 pagini intregi in manualul de utilizare.

EXPORT x$ — salveaza tabela de simboluri pe banda ,disc sau microdrive.

IMPORT x$ — incarca tabela de simboluri salvata cu ajutorul instructiunii EXPORT.

REDUCE — functioneaza asemanator lui CLEAR , dar nu sterge valoarea etichetelor protejate prin clauza speciala CARGO.

PSEUDOCOMENZI ALE ASAMBLORULUI

ORG x — stabileste x ca fiind valoarea numaratorului de locatii ($). In acest fel ,programul asamblat va functiona doar de la aceasta adresa. (Implicit $=RAMTOP).

PUT x — stabileste x ca fiind valoarea numaratorului de adrese (.). Codul obiect va fi stocat la aceasta adresa ,ceea ce nu inseamna ca va si functiona in aceasta pozitie. Folosind PUT $,numaratorul de adrese va lua valoarea numaratorului de locatii. (Implicit .=RAMTOP). La SPECTRUM 128 se poate selecta si pagina RAM in care va fi depus codul obiect.

DEFB parametri — stocheaza octeti in codul obiect. Spre deosebire de GENS ,parametrii pot fi de orice tip (de ex: DEFB 22, "VBS",1988). Comanda DEFB poate fi prescurtata prin DB.

DEFM parametri — aceasta comanda a fost introdusa pentru compatibilitate cu GENS. Acelasi efect se poate obtine cu DEFB.

DEFW parametri — stocheaza numere pe 16 biti in codul obiect. Poate fi prescurtata prin DW.

DEFS x — rezerva un spatiu de x octeti in codul obiect. Se prescurteaza prin DS.

EQU x — defineste valoarea numerica a unei etichete.

eticheta: DEFL x — modifica valoarea unei etichete deja definite. Iata in continuare un exemplu de utilizare a acestei comenzi:

```
count: EQU 1
       *WHILE count<=15
       DEFB 2*count-1
count: DEFL count+1
       *ENDW
```

Aceasta secventa construieste tabela primelor 15 numere impare ,utilizind un ciclu WHILE...ENDW.
Comanda DEFL poate fi prescurtata prin DL.

COND x... ...ELSE... ...ENDC — daca este indeplinita conditia x ,asamblarea continua. Daca nu,asamblarea se executa de dupa comanda ELSE. Sfirsitul secventei de asamblare conditionata se marcheaza cu ENDC.
Instructiunea ELSE poate lipsi ,folosindu-se doar constructia COND...ENDC.Constructia COND x...ELSE...ENDC este identica in functionare cu IF x...ELSE...END din GENS.

eticheta :CARGO — foloseste la protejarea etichetei de stergerea prin comanda REDUCE.

PSEUDOOPTIUNI ALE ASAMBLORULUI

Pseudooptiunile asamblorului conteaza doar la asamblare si nu au nici un efect asupra codului obiect. Se introduc in listingul fisierului sursa si asemeni pseudooptiunilor GENS ,incep cu o steluta (*). Litera f specifica in continuare un flag (ON sau OFF).

*SCREEN f — permite sau nu accesul la ecran in timpul asamblarii. Implicit ON.

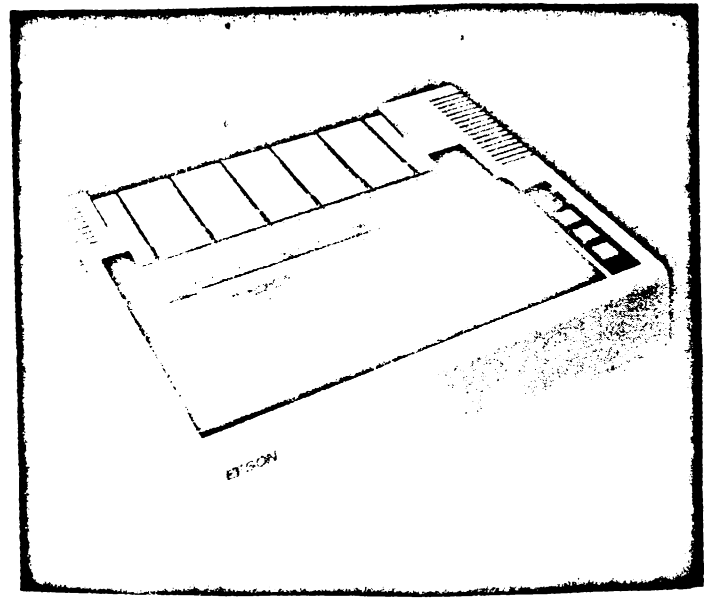
*PRINTER f — permite sau nu iesirea la imprimanta in timpul asamblarii. Implicit OFF.

*LIST f — provoaca sau nu listarea programului pe ecran in timpul asamblarii. Implicit OFF.

*LLIST f — ca si *LIST ,dar la imprimanta.

*FORM — trimite un caracter 'form feed' la imprimanta.

*TITLE x$ [,x] — permite scrierea de titluri pe paginile hirtiei de la imprimanta ,in timpul asamblarii. x$ reprezinta titlul ,iar x numarul paginii.



*MACLIST f — permite sau nu prezentarea codului care se genereaza cu ajutorul macroinstructiunilor. Implicit OFF.

*INCLUDE x$ — face asamblarea de pe banda/disc/microdrive. Plaseaza fisierul x$ pe pozitia data in program. Are acelasi e-

fect ca si *F la GENS. La SPECTRUM 128 ,aceasta optiune lucreaza si cu RAMDISC-ul.

***OPENOUT x$** face asamblarea pe banda/disc/microdrive. Salveaza codul obiect in blocuri scurte ,in timpul asamblarii ,pe dispozitivul curent de iesire. Cu ajutorul combinatiei INCLUDE/OPENOUT pe disc sau microdrive ,se pot asambla simplu programe foarte lungi.

***CLOSEOUT** opreste trimiterea codului obiect catre dispozitivul de iesire.

***PROMPTS f** permite sau nu tiparirea mesajelor (de ex: 'start tape') in timpul asamblarii. Implicit ON. Numai pentru SPECTRUM.

***COUNT f** porneste sau opreste afisarea numaratorului de linii in timpul asamblarii . Implicit OFF.

***REPORT f** permite sau nu tiparirea mesajelor de eroare in timpul asamblarii. Implicit ON. *REPORT OFF se foloseste uzual la utilizarea comenzii ASSEML.

***CODE f** permite sau nu generarea codului obiect. Implicit ON.

***PRINT x$** tipareste stringul x$ in timpul asamblarii.

***PAUSE** la intilnirea acestei optiuni ,asamblorul asteapta apasarea unei taste.

***WHILE x** inceputul buclelor WHILE la asamblare . Instructiunile dintre *WHILE si *ENDW vor fi asamblate atit timp cit conditia x este adevarata.

***ENDW** sfirsitul buclelor WHILE la asamblare

***REPEAT** inceputul buclelor REPEAT-UNTIL la asamblare.

***UNTIL x** sfirsitul buclelor REPEAT-UNTIL la asamblare. Instructiunile dintre *REPEAT si *UNTIL vor fi asamblate pina cind conditia x va fi adevarata.

PSEUDOCOMPILATORUL PHOENIX

Pseudocompilatorul PHOENIX reprezinta cea mai importanta inovatie a pachetului LASER.

Este vorba despre un compilator al unui limbaj foarte simplu :PHOENIX (intrucit autorul nu a mai auzit despre acest limbaj ,presupune ca ar fi vorba despre o inventie a programatorilor firmei OASIS). Limbajul are posibilitati relativ mici ,dar faptul ca instructiunile sale pot fi amestecate cu instructiuni Z 80 ,ii confera o forta deosebita si permite atingerea unor rezultate remarcabile.

Impreuna cu LASER se primesc programe demonstrative pentru desenarea rapida a cercurilor si elipselor si calculul numerelor prime cu ciurul lui Eratostene. Programele demonstrative sint scrise in doua versiuni : in cod masina pur si intr-o combinatie 90 % PHOENIX si 10 % limbaj de asamblare. Aceasta a doua versiune se deosebeste in viteza foarte putin de prima ,iar lungimea codului obiect este cu aproximativ 15 % mai mare decit in cazul codului masina pur.

PHOENIX poseda patru tipuri de variabile numerice si sir : INT , CHAR , PINT si PCHAR. Prefixul 'P' deriva de la POINTER. Variabilele numerice si sir se declara cu ajutorul comenzii #DS (de altfel ,toate comenzile PHOENIX sint precedate de caracterul '#' ,de unde si denumirea de HASH EXTENSION). Aceasta comanda se utilizeaza in forma :

```
identificator:#DS  tip,x
```

De exemplu :

```
num: #DS  INT,15
```

delara un sir de 15 numere intregi ,cu numele 'num'. Elementele sale sint : num [1] , num [2] samd. Bineinteles ,indexurile pot fi expresii numerice.

Comanda :

```
numar: #DS  INT,1
```

defineste o variabila intreaga obisnuita ,cu numele 'numar'. Comanda :

```
cuvint: #DS  CHAR,40
```

declara un string cu numele 'cuvint' si lungimea de 40 de caractere. Stringurile nu pot fi apelate global ,ci doar caracter cu caracter (de ex: cuvint [5]) .

In mod analog se declara si variabilele de tip POINTER.

O comanda foarte asemanatoare comenzii #DS este comanda #DI care declara variabila si totodata ii asociaza si valoarea. Declaratia :

```
xcor: #DI  INT,255
```

este echivalenta in BASIC cu :

```
LET  xcor = 255   .
```

In mod analog pot fi declarate si stringuri :

```
text: #DI CHAR,"Video Byte Studios"     .
```

Compilarea expresiilor aritmetice

Calculul valorilor expresiilor se executa in PHOENIX cu ajutorul comenzilor #DSE si #DUE. Prima realizeaza calculul in notatie cu semn ,iar cea de-a doua ,in notatie fara semn. Pe linga operatorii expresionali enuntati ,PHOENIX mai admite si operatori noi. Dintre acestia ,cel mai important este operatorul de asignare '=' (a nu se confunda cu '?='). Acest operator se evalueaza de la dreapta la stinga si are cea mai mica prioritate (1) .

Comanda :

```
#DUE x = y * y + 5
```

este echivalenta cu comanda BASIC :

```
LET x = y * y + 5     .
```

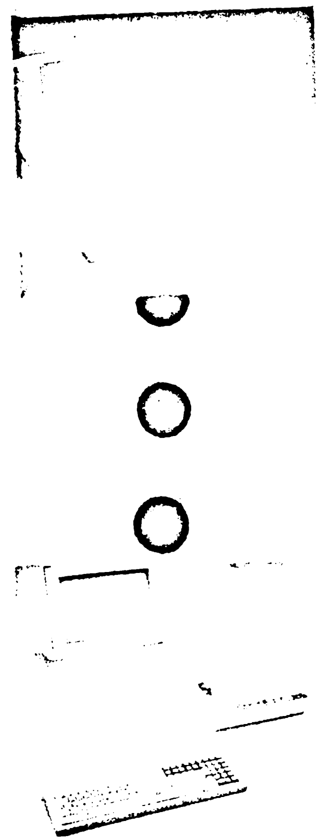
Expresiile se calculeaza intr-un mod foarte asemanator limbajului C ,fiind posibile si combinatii ca :

```
#DUE x = y = 0   ,
```

care este echivalenta cu :

```
LET y = 0
LET x = 0     ,
```

sau chiar :



```
#DUE x = (y = z * z) + z - 15  ,
```

care ar putea fi scrisa in BASIC astfel :

```
LET y = z * z
LET x = y + z - 15     .
```

Comanda :

```
#DUE x = y ?= z
```

are in BASIC urmatorul echivalent :

```
LET x = (y = z)
```

Comenzile #DSE si #DUE pot fi utilizate si fara operatorul '=' , caz in care rezultatul va fi introdus in registrul pereche HL al procesorului. Pentru a simula comanda BASIC :

```
DPOKE 30000,x+y
```

vom proceda astfel :

```
#DUE x+y
LD   (30000),HL      ,
```

in timp ce pentru a obtine efectul comenzii :

```
POKE 30000,x
```

va trebui sa folosim o secventa :

```
#DUE x
LD   A,L
LD   (30000),A      .
```

Expresiile pot contine variabile amestecate ,din toate cele patru tipuri. Programul :

```
numar:  #DS  INT,1
string: #DI  CHAR,"ab"
        #DUE numar = string [1] + 2 * string [2] - 5
```

este echivalent cu linia BASIC :

```
LET numar = CODE "a" + 2 * CODE "b" - 5       .
```

Doi operatori noi imprumutati din limbajul C sint :'++' si '--'. Ei realizeaza incrementarea ,respectiv decrementarea ope-

randului care ii succede. De exemplu :

```
#DUE x = ++y
```

este echivalent cu :

```
LET y = y + 1
LET x = y
```

iar :

```
#DUE --x
```

este echivalent cu :

```
LET x= x-1
```

si noua valoare a lui x se transfera in registrul HL.
Operatorul unar '&' semnifica 'adresa lui...'. De exemplu :

```
#DUE  ptr = &cnt
```

confera variabilei ptr ,care trebuie sa fie de tip POINTER ,adresa variabilei cnt. Constructiile de tipul :

```
#DUE byte = *count
```

sint posibile doar daca count este o variabila de tip POINTER si se refera la continutul variabilei pe care o vectorizeaza variabila POINTER.

Variabilele de tipul PINT se deosebesc de cele de tipul PCHAR doar dupa actiunea operatorilor inrementali si decrementali (sa nu uitam ca o variabila de tip INT se retine pe doi octeti ,in timp ce o variabila de tip CHAR se retine pe un singur octet).

Pentru transferul valorilor din registre in variabile poate fi folosit urmatorul truc :

```
work: DEFS 2
var:  #DS  INT,1
      LD   (work),HL
      #DUE var = *work
```

Secventa de mai sus simuleaza comanda :

```
LET var = HL
```

FUNCTII , CONDITII SI CICLURI

PHOENIX permite si definirea functiilor. Acestea se declara cu ajutorul comenzii #FNC ,in sintaxa :

```
identificator: #FNC  tip
```

Parametrii se declara prin comanda #PRM.
Inceputul definitiei functiei se marcheaza prin #BEGIN ,iar sfirsitul prin #END. Rezultatul final al functiei este cel continut in registrul HL (sau L daca functia este de tip CHAR).

Iata in continuare si un exemplu de definire a unei functii :

```
square: #FNC  INT
x:      #PRM  INT
y:      #PRM  INT
        #BEGIN
        #DUE x * x + y * y
        #END
```

Functia astfel definita poate fi apelata prin :

```
#DUE z - square (2,3)          sau
#DUE square (10,z+3) - 9       samd.
```

Inainte de comanda #BEGIN pot fi definite variabile locale (prin intermediul lui #DS). Parametrii functiei sint de asemeni variabile locale ,astfel ca functiile pot fi apelate recursiv.

Functiile fara parametri se apeleaza prin doua paranteze fara continut. De exemplu :

```
#DUE  t = time ()
```

Functia poate sa nu dea nici un rezultat ,caz in care valoarea registrului HL este imprevizibila.

Comenzile #IF x , #ELSE si #ENDIF sint clare si nu necesita explicatii. In exemplul care urmeaza poate fi urmarita actiunea acestor comenzi :

```
maxim: #FNC  INT
x:     #PRM  INT
y:     #PRM  INT
       #BEGIN
       #IF   x > y
       #DUE  x
       #ELSE
```

```
#DUE  y
#ENDIF
#END
```

Dupa definirea acestei functii ,comanda :

```
#DUE n = maxim (5,9)
```

asociaza variabilei n valoarea 9.

De asemeni ,comenzile #WHILE x ,#ENDW ,#REPEAT si #UNTIL x sint binecunoscute.

Urmatorul exemplu defineste functia putere :

```
power: #FNC  INT
x:     #PRM  INT
y:     #PRM  INT
local: #DS   INT,1
       #BEGIN
       #DUE  local = 1
       #REPEAT
       #DUE  local = local * x
       #DUE  --y
       #UNTIL y ?= 0
       #DUE  local
       #END
```

Urmatoarea functie scrie un caracter pe ecran si nu transmite nici un rezultat :

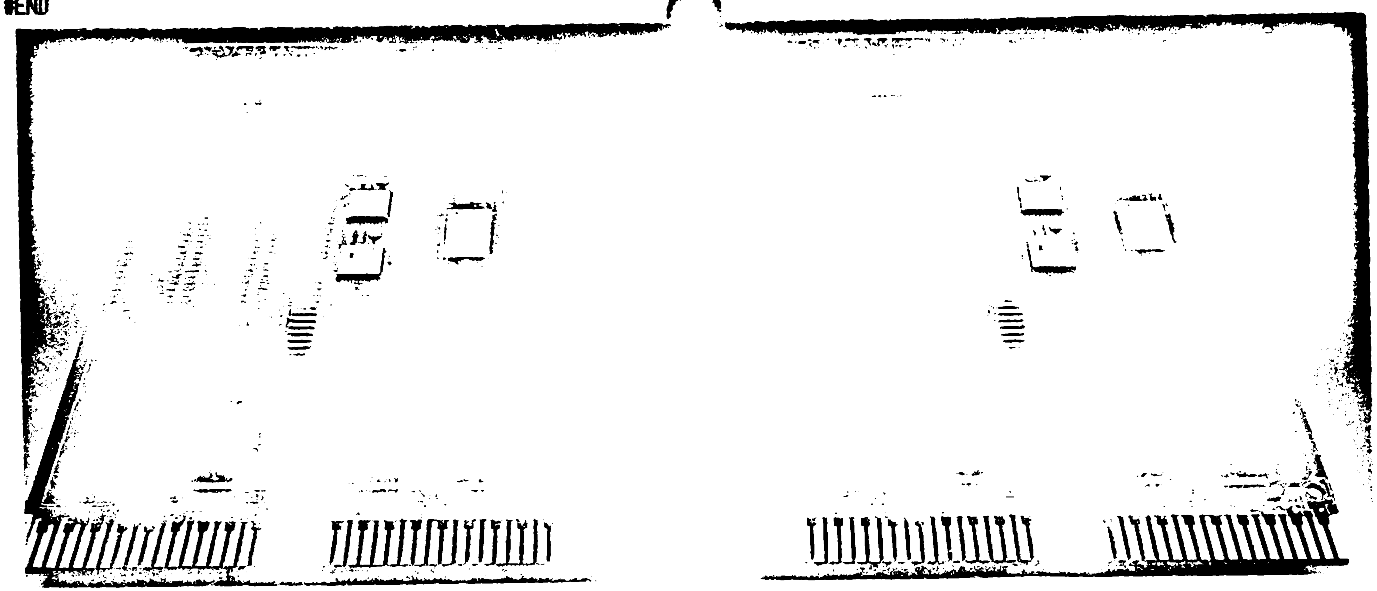
```
chrout: #FNC INT  ;tipul este neesential,functia ne-
                  ;transmitind nici un rezultat
code:   #PRM CHAR
        #BEGIN
        LD   A,2
        CAll 1601H
        #DSE code
        LD   A,L
        RST  16
        #END
```

Comanda #RETURN foloseste la intoarcerea din evaluarea functiei ,oriunde in interiorul unei constructii BEGIN...END.

ALTE INSTRUCTIUNI

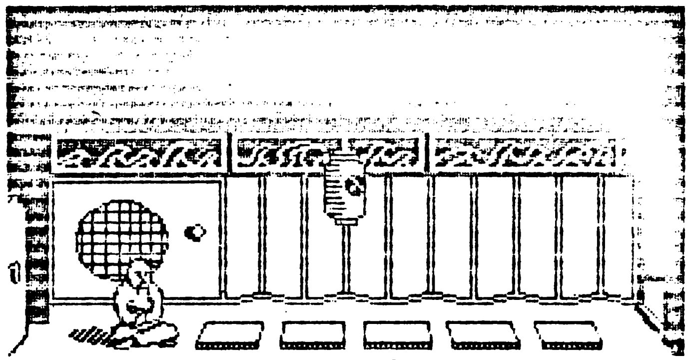
In final au mai ramas de discutat comenzile #LIB si #STACK.

Comanda #LIB trebuie obligatoriu sa se gaseasca la sfirsitul unui program care utilizeaza instructiuni PHOENIX. Aceasta instructiune introduce in codul obiect rutine pentru executarea comenzilor PHOENIX. Daca se omite aceasta comanda ,se obtine un



gen de p-code si orice incercare de a lansa programul are un deznodamint fatal.

Comanda #STACK [x] pregateste stiva aritmetica pentru uti-





lizarea functiilor si de obicei se introduce la inceputul programelor. Aceasta instructiune nu e indispensabila in toate cazurile ,dar in general este dezirabila. Ea genereaza urmatorul cod :

```
LD   IX,0
ADD  IX,SP
DEC  SP
```

Daca se introduce si parametrul x ,la inceputul secventei de mai sus se adauga :

```
LD   SP,x
```

,astfel incit intoarcerea in BASIC trebuie facuta cu atentie.

Daca se foloseste instructiunea #STACK fara parametri ,intoarcerea in BASIC trebuie facuta prin :

```
INC  SP
RET
```

Mai trebuie avuta in vedere o greseala uzuala. De exemplu , avind programul :

```
        ORG  25500
        PUT  $
var:    #DI  INT,100
start:  #DUE var
        LD   (23728),HL
        RET
```

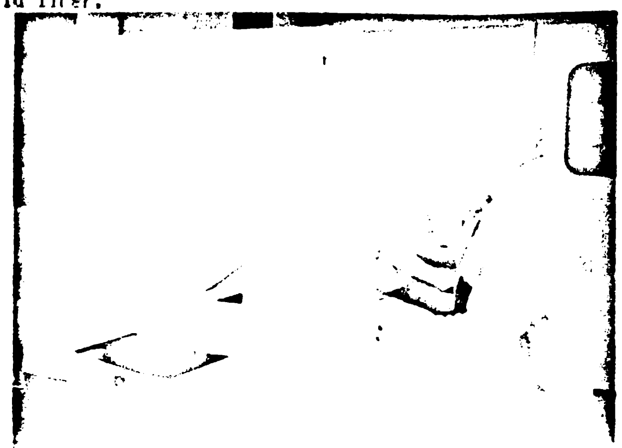
,acest program nu trebuie in nici un caz lansat de la adresa 25500 ,ci de la valoarea etichetei 'start' (aici 25502 ,ceea ce se poate afla prin UPRINT start). Cea mai indicata este lansarea programului direct din editor ,prin EXECUTE start.

IN INCHEIERE

Cele de mai sus reprezinta o scurta descriere a posibilitatilor pachetului LASER.

Ar mai fi citeva cuvinte de spus despre MPAFNCSPHN si TRANS ,programe care se livreaza impreuna cu asamblorul. MPAFNCSPHN contine definirea unor functii PHOENIX care permit lucrul in FP (doar adunare ,scadere , inmultire si impartire). TRANS transforma blocuri salvate prin OPENOUT si SAVE intr-un format care permite incarcarea lor din BASIC.

In final ,iata si un bug - nu salvati nimic din LASER pe microdrive ,daca nu sinteti sigur ca pe cartridge este suficient spatiu liber.



MIRCEA THEODORESCU

LOGO

I. GRAMATICA LIMBAJULUI

Elementele fundamentale ale limbajului LOGO sint procedurile si cuvintele; cuvintele se definesc ca serii de caractere alfanumerice, fiecare caracter al cuvintului constituind un element al acestuia. Cuvintele se disting de numele procedurilor prin utilizarea ghilimelelor in fata lor, spre exemplu: "CARTE (intre " si carte nu trebuie lasat spatiu). Exemplu de utilizare al unei proceduri: PRINT "carte (scrie cuvintul carte). Numerele sint cuvinte iar pot fi folosite fara ghilimele in fata lor ca si cuvintele predefinite TRUE si FALSE.

OBIECTE: sint cuvinte sau liste utilizate ca intrari si iesiri pentru proceduri; listele sint o serie de cuvinte si liste, deci apare o structura imbricata a listelor. Lista se incadreaza intre paranteze drepte si componentele din interior se separa prin spatii intre ele.

EXEMPLE DE LISTE:

1. [] -lista vida
2. [radio frigider masa] -are 3 elemente
3. [[radio frigider] [masa]] -are 2 liste

SEPARATORI: SPACE, [,], (,), =, <, >, +, -, *, /

INTRARI: unele proceduri au nevoie de obiecte la intrare pentru a functiona; aceste obiecte sint date fie la apelarea procedurii, fie ca iesiri ale altor proceduri.

Procedurile se impart in doua grupe:
1. PRIMITIVE
2. DEFINITE DE UTILIZATOR

SINTAXA DE DEFINIRE A UNEI PROCEDURI:

```
TO nume :obiect   (linia titlu)
   .........
END
```

Procedurile PRIMITIVE exista deja in sistem.

In procedurile care necesita intrari acestea se specifica la inceputul definirii procedurii in linia titlu.

exemplu: TO scrie :telefon

-telefon va fi obiectul de intrare pentru procedura cu numele scrie; pot apare mai multe intrari folosind ca delimitatori spatiile intre intrari.

Daca un identificator este folosit fara sa fie precedat de nimic, LOGO interpreteaza aceasta ca o chemare a unei proceduri cu acel nume.

Daca identificatorul este precedat de simbolul
" -atunci este cuvint
: -atunci este obiect
[-este intr-o lista

(-atunci permite utilizarea mai multor intrari la o primitiva; paranteza este folosita si in operatiile aritmetice.

Dupa modul in care lucreaza, procedurile se pot clasifica in COMENZI si OPERATII. COMENZILE spre deosebire de OPERATII nu au iesiri, deci nu au ca rezultat al executiei o valoare. OPERATIILE au la iesiri valori care trebuie sa fie insa intrari la rindul lor pentru alte proceduri, lantul oprindu-se intr-o comanda.

```
   exemplu: TO aduna5 :x
            :x + 5
            END
```

```
          ?aduna5 6
   mesaj: you don't say what to do with 11
```

```
   exemplu corect: TO aduna5 :x
                   OUTPUT :x + 5
                   END
```

```
          ?PRINT aduna5 6
   apare pe ecran: 11
```

Valoarea de iesire a operatiei aduna5, care este 11 serveste ca intrare pentru comanda PRINT care este o primitiva.
Observatie: o comanda nu poate fi folosita ca intrare pentru alta comanda deoarece nu poseda iesire, deci nu pot fi inlantuite doua comenzi.
exemplu incorect: PRINT FD 100 ; pentru a deplasa turtle o suta de pasi .
mesaj: 'fd does not output to print'

VARIABILE:

Variabilele pot fi create utilizind procedura MAKE. exemplu: MAKE "a 1
Valoarea variabilei se obtine astfel: PRINT :a
Variabilele pot tine locul la o varietate de tipuri de date.

```
   exemplu:   MAKE "a [1 2 3]
      sau     MAKE "a [rosu 10 galben grad]
```

Variabilele, in functie de regiunea pe care actioneaza sint locale sau globale.
a) Variabilele create cu procedura MAKE sint globale, valabile pentru orice procedura si exista si dupa executia procedurilor.
b) Variabilele definite in linia titlu a unei proceduri sint locale pentru procedura respectiva si subprocedurile continute de aceasta, existind numai pe parcursul ei.

O linie LOGO poate contine pina la 242 de caractere. Continuarea se realizeaza automat prin semnul ! ; linia se termina apasind tasta ENTER.

REGULI DE FORMARE A INSTRUCTIUNILOR:

1. Daca apare un nume de procedura trebuie sa observam ce este: comanda sau operatie si cite intrari are.
2. Prima procedura a unei linii LOGO trebuie sa fie o comanda.
3. Operatiile au iesiri care servesc ca intrari in alte proceduri (lantul se sfirseste intr-o comanda).
4. Toate intrarile din proceduri trebuie folosite

---

exemplu:

```
?PRINT SENTENCE [I AM] WORD BUTLAST :cuv "IER

                PRINT
                  |
                  |
             |----|----|
             |         |
             |         |
          [I AM]     WORD
                       |
                  |----|----|
                  |         |
               BUTLAST     "IER
                  |
                 CUV
```

Daca valoare variabilei CUV este HAPPY atunci pe ecran va apare tiparit: I AM HAPPIER.

PRIORITATEA operatiilor aritmetice este: / * - +.
Observatie: print random 4 + 7 este acelasi lucru

cu print random II.

ECRAN, MODURI SI PROMPTERE:

-Ecranul de text are 22 de linii si ultimile doua linii pentru tiparit mesaje ca si in BASIC;
-Ecranul de grafica are 24 linii.

a) Modul direct de lucru: fiecare instructiune este interpretata si executata imediat; prompterul este '?'
b) Modul TO: este utilizat pentru scrierea procedurilor; prompterul este '>'
c) Modul editare: este utilizat pentru creare sau modificare proceduri; prompterul este flashing.

RECURSIVITATE:

Acest fenomen apare atunci cind o procedura se apeleaza pe ea insasi. Fiacare autoapelare a procedurii formeaza o recursie descendenta daca nr. procedurii care se apeleaza creste, in mod direct fiecare iesire a procedurii fiind intrare la o comanda la fiecare pas; recursia este ascendenta daca controlul este preluat crescator prin inlantuirea procedurii, adica iesirile vor constitui intrari in comenzi, in ordine inversa fata de celalalt tip de recursie, de la coada spre cap. Urmeaza apoi comenzile STOP si END.

```
exemplu: RECURSIE DESCENDENTA
    TO dubleaza :Start
        IF :Start > 50 [ STOP ]
        PR :Start
        dubleaza :Start * 2
    END
```

```
      ? dubleaza 5
mesaj:   5
        10
        20
        40
```

```
exemplu: RECURSIE ASCENDENTA
    TO tripleaza :Start
        IF :Start < 80 [ tripleaza :Start * 3]
        PR :Start
    END
```

```
      ? tripleaza 5
mesaj:    135
           45
           15
            5
```

```
exemplu de recursie descendenta:
    TO numara :N
        IF :N = 0 [ STOP ]
        PR :N
        numara :N - 1
    END
```

```
      ? numara 3
mesaj:  3
        2
        1
```

```
exemplu de recursie ascendenta:
    TO enumara :N
        IF :N = 0 [ STOP ]
        enumara :N - 1
        PR :N
    END
```

```
      ? enumara 3
mesaj:  1
        2
        3
```

Prima procedura (numara) este executata in mod direct fiecare iesire a procedurii fiind la fiecare pas intrare pentru comanda PRINT.

A doua procedura (enumara) functioneaza ceva mai complicat:

i) prima instructiune verifica daca valoarea lui N este 0;

ii) a doua introduce N - 1 in aceeasi procedura enumara (daca valoarea lui N era mai mare ca zero);

iii) instructiunea PRINT :N se executa la momentul in care in procedura enumara, N intra cu valoarea 0, dar nu se incheie executia deoarece N a avut mai multe valori care asteapta in memorie spre a fi introduse in comenzi ca spre exemplu in cazul de fata PRINT. LOGO in cazul de fata tipareste prima data 1.

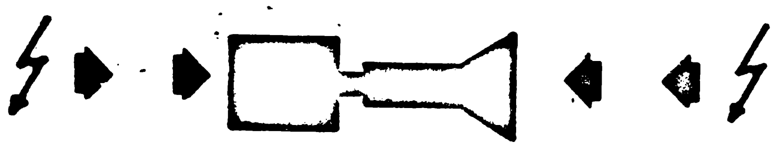
iv) urmatoarea instructiune este END dar nu se executa deoarece mai sint valori ale lui N in memorie, deci LOGO da un pas inapoi in cadrul inlantuirii recursive, unde "enumara" avea valoarea lui N:2, si se tipareste 2, dupa care se reia rationamentul pina se termina valorile lui N. Aceasta se intimpla ori de cite ori mai sint de executat instructiuni, deci daca procedura este apelata undeva in mijlocul procedurii.

IESIREA DIN LIMBAJUL LOGO:

Daca se doreste iesirea din program fara a se reseta calculatorul, se tasteaza comanda BYE si se intra din nou in interpretorul de BASIC.

Pentru a reintra in LOGO se tasteaza RUN si se intra fara a modifica ce s-a lucrat.

*
* *



II. TURTLE

++++++++++*

Turtle este triunghiul care apare pe ecran cind se lucreaza cu grafica limbajului. Ori de cite ori se apeleaza o primitiva legata de miscarea lui TURTLE, apare triunghiul si se trece in ecranul de grafica. Dispare daca se trece in modul de editare sau text. Ecranul de grafica este 256*175 daca nu este modificat de utilizator

primitivele utilizate pentru TURTLE sint:

BACK (BK) n - TURTLE se misca n pasi inapoi fara a-si modifica directia;

BACKROUND (BG) -intoarce un numar intre 0 si 7 reprezentind culoarea hirtiei;

CLEAN -sterge ecranul fara a schimba pozitia lui TURTLE;

CLEARSCREEN (CS) -sterge ecranul si muta TURTLE in pozitia initiala in centrul ecranului;

DOT [x y] -pune un punct la pozitia specificata de x si y. TURTLE nu se misca ;

FENCE -miscarile lui TURTLE sint limitate la marginile ecranului. Daca acestea sint depasite prin instructiuni se trimite mesajul: "turtle out of bounds" ;

FORWARD (FD) n -TURTLE se misca n pasi inainte (sau inapoi daca n este negativ);

HEADING -intoarce un numar intre 0 si 359 corespunzator orientarii lui TURTLE;

HIDETURTLE (HT) -sterge triunghiul de pe ecran, crescind viteza de desenare;

HOME -muta TURTLE in pozitia initiala, in centrul ecranului si daca creionul era jos se trage o linie din pozitia anterioara pina in centrul ecranului. Orientarea devine 0;

LEFT (LT) n -- TURTLE se roteste cu n grade la stinga;

PENCOLOUR (PC) -intoarce un numar intre 0 si 7 reprezentind culoarea creionului;

PENDOWN (PD) -lasa creionul jos astfel incit daca TURTLE se misca apar linii pe ecran

PENERASE (PE) -sterge liniile pe deasupra carora trece TURTLE;

PENREVERSE (PX) -sterge acolo unde sint linii si trage linii acolo unde nu sint;

PENUP (PU) -daca TURTLE se misca nu este trasa nici o linie;

POSITION (POS) -intoarce pozitia lui TURTLE in coordonate ecran;

RIGHT (RT) n - se roteste cu n grade la dreapta;

SCRUNCH -intoarce raportul [x y] dintre coordonatele ecranului, initial 100 100;

SETBG n -seteaza culoarea hirtiei;

SETBORDER (SETBR) n - seteaza culoare border;

SETHEADING (SETH) n - seteaza orientarea lui TURTLE;

SETPC n - seteaza culoare creion;

SETPOS [x y] -TURTLE se deplaseaza pina la pozitia indicata lasind eventual o dira

SETSCRUNCH [x y] - seteaza raportul x si y;

SETX n -deplaseaza TURTLE pe x lasind ordonata neschimbata;

SETY n - la fel ca SETX n pe y;

SHOWNP -intoarce valoarea de adevar TRUE daca TURTLE este in SHOWTURTLE sau FALSE altfel;

SHOWTURTLE (ST) - face TURTLE vizibil;

TOWARDS [x y] -intoarce orientarea lui TURTLE pe care ar trebui sa o aiba in acea pozitie [x y]. TURTLE este neafectat;

WINDOW -este opusa lui FENCE, deci TURTLE poate parasi marginea ecranului. Se poate deplasa maxim 32767 pasi din centru;

WRAP - acelasi lucru cu WINDOW;

XCOR - intoarce coordonata x a pozitiei curente;

YCOR - intoarce coordonata y a pozitiei curente.

III. CUVINTE SI LISTE

Cuvintele si listele constituie obiecte.Un cuvint este o serie de caractere alfanumerice exceptind SPACE-ul, fiecare caracter fiind un element al cuvintului. Limitele cuvintelor sint marcate prin SPACE sau sint precedate de caracterele : si " fara a lasa SPACE intre cuvint si simbol. " urmat de SPACE reprezinta cuvintul vid.

Pentru a introduce in cuvinte si caracterele []()+-*/ trebuie folosit in fata cuvintelor / . Acest slash spune limbajului ca semnele respective trebuie tratate ca elemente de cuvint si nu altfel.

Listele sint compuse din cuvinte si alte liste, ceea ce duce la formarea unei structuri imbricate.

PRIMITIVELE CARE LUCREAZA CU CUVINTE SI LISTE:

ASCII "caracter (daca este cifra nu trebuie ") - intoarce codul ascii pentru caracterul introdus;

BUTFIRST obiect (BF obiect) -intoarce tot, cu exceptia primului caracter al obiectului specificat (obiectul fiind cuvint sau lista). BF pentru un obiect vid este o eroare;

BUTLAST obiect (BL obiect) -analog cu BUTFIRST dar pentru ultimul caracter;

CHAR n -intoarce caracterul al carui cod ASCII este n intre 32 si 165;

COUNT obiect -intoarce numarul de elemente ale obiectului specificat ;

EMPTYP obiect -intoarce valoarea TRUE daca obiectul este vid sau FALSE altfel;

EQUALP obiect1 obiectn -intoarce valoarea TRUE daca obiect1 si obiect2 sint numere egale sau cuvinte identice sau liste identice; FALSE altfel;

FIRST obiect -intoarce primul element al obiectului care poate fi cuvint sau lista;

FPUT obiect lista -intoarce o noua lista in care primul element va fi obiectul specificat;

ITEM n -intoarce al n-lea element dintr-o lista data initial;

LAST obiect -intoarce ultimul element al obiectului specificat;
LIST obiect1......obiectn -intoarce o lista in care elementele sint obiectele specificate;
LISTSP obiect -intoarce valoarea TRUE daca obiectul este o lista, altfel FALSE. O lista vida este tratata ca un cuvint;
LASTPUT (LPUT) obiect lista -intoarce o noua lista in care ultimul element este obiectul specificat;
MEMBERP obiect lista -intoarce valoarea TRUE daca obiectul apartine listei, FALSE altfel;
SENTENCE (SE) obiect1...obiectN -intoarce o lista formata din obiectele de intrare;
WORD cuvint1...cuvintN -intoarce un cuvint format prin concatenarea cuvintelor initiale;
WORDP obiect -intoarce TRUE daca obiectul este cuvint, sau FALSE altfel.

---

exemple de utilizare:

operatie	intrare1	intrare2	iesire
FPUT	"TELE	"FON	[TELEFON]
LIST	"TELE	"FON	[TELEFON]
LPUT	"TELE	"FON	[TELEFON]
SE	"TELE	"FON	[TELEFON]

IV. VARIABILE

Variabilele pot fi create utilizind procedura MAKE sau la intrarile unei proceduri, in momentul definirii. Este posibil de a se atasa o valoare unei variabile care la rindul ei sa fie valoare pentru alta variabila, construind astfel o structura arborescenta.

PRIMITIVE:

MAKE nume obiect -ataseaza valoarea 'obiect' variabilei 'nume';

```
exemplu:   MAKE "ANIMAL "PISICA
           PR :PISICA

mesaj:     PISICA

           MAKE :ANIMAL "PISOIAS
           PR :PISICA

Mesaj:     PISOIAS
```

NAMEP obiect -intoarce valoarea TRUE daca obiectul are o valoare altfel FALSE;
THING nume -intoarce continutul variabilei la fel ca si :nume;

V. OPERATII ARITMETICE

Adunarea, scaderea, inmultirea si impartirea pot fi utilizate ca simboluri: + - * / sau in formele prefixate: SUM, DIV sau PRODUCT urmate de cele doua intrari corespunzatoare. Mai este folosita si primitiva EQUALP in operatiile aritmetice

PRIMITIVELE OPERATIILOR ARITMETICE:

ARCCOS n -ca si in BASIC;

ARCCOT n -----""----- ;

ARCSIN n -----""----- ;

ARCTAN n -----""----- ;

COSINE (COS) n --""-- ;

COTANGENT (COT) n -""- ;

DIV a b -intoarce citul impartirii lui a prin b;

INT n -intoarce partea intreaga din n;

PRODUCT a......n

RANDOM n -daca n este un intreg pozitiv, intoarce un numar intre 0 si n-1;
REMAINDER a b -intoarce restul impartirii lui a la b;
ROUND n -intoarce valoarea lui n rotunjita la cel mai apropiat intreg;
SINE (SIN) n - folosit ca si in BASIC;

SQRT n -radacina patrata a lui n;

SUM a.....n -intoarce suma celor n termeni;

TANGENT (TAN) n -folosita ca si in BASIC;

Trebuie avut grija la diferenta intre semnul - si operatia pe care o defineste semnul. Semnul fara SPACE intre el si obiect defineste un numar negativ, operatia de scadere obtinindu-se cu SPACE intre operanzi.
Mai exista operatiile > < si =.
Spre exemplu: a < b intoarce valoarea TRUE daca este adevarata expresia altfel FALSE.

VI. DEFINIRE SI EDITARE

++++++++++++++++++++++

EDIT (ED) nume-procedura -se intra in modul editare la procedura sau lista de proceduri specificate. ED fara nimic aduce un editor gol.

CONTROL EDITOR: -deplasarile se fac cu cursorul;
- -Extended-mode + B deplaseaza cursorul la inceputul textului;
- -Extended-mode + E deplaseaza cursorul la sfirsitul textului;
- -Emode + Y sterge linia de la pozitia cursorului in jos si o salveaza;
- -Emode + R introduce linia salvata la pozitia curenta a cursorului;
- -Symbolshift + S opreste scroll, apasarea oricarei taste reia scroll-ul;
- -Emode + P muta cursorul la pagina anterioara
- -Emode + N muta cursorul la urmatoarea pagina
- -Emode + C introduce modificarile facute si se retrece in modul direct;
- -BREAK ignora modificarile efectuate si se retrece in modul direct;

primitiva EDNS :
EDNS nume
EDNS [nume lista] -editeaza numele obiectelor si valorile lor din lista specificata. Fara nici un parametru listeaza toate variabilele si valorile lor.

TO nume intrare1....intraren -se intra in modul definire procedura; pe ultima linie trebuie sa fie numai comanda END.

VII.EXPRESII CONDITIONALE SI PRELUAREA CONTROLULUI

++++++++++++++++++++++++++++

Instructiunile de conditie fac sa se execute o anumita instructiune daca este indeplinita o anumita conditie;
Instructiunile de repetitie fac sa se repete o lista de instructiuni de mai multe ori;
Instructiunea STOP opreste executia procedurii curente si se preia controlul de catre procedura imediat superioara in inlantuire.

SINTAXA:

IF pred listainstr1 listainstr2 -If-ul testeaza valoarea de adevar a predicatului dupa care daca valoarea este TRUE atunci se executa lista de instructiuni 1 altfel se executa cealalta lista; listele pot sa nu fie prezente amindoua in acelasi timp si atunci avem IF partial.

```
exemplu:
   TO alege
      IF 0 = RANDOM 3 [OP "yes] [OP "no]
   END
   ?PRINT alege
    no
```

VIII. OPERATII LOGICE

+++++++++++++++++++++

AND pred1...predn -daca AND are mai mult de doua intrari se folosesc parantezele () in jurul instructiunii si intrarilor - intoarce o valoare logica TRUE sau FALSE;

NOT pred -neaga un predicat;

OR pred1....predn - la fel ca si AND

IX. CUVINTE EXTERIOARE

++++++++++++++++++++++

KEYP -intoarce TRUE daca o tasta sau o combinatie de taste a fost apasata, altfel FALSE;

PRINT (PR) obiect -tipareste continut obiect;

READCHAR (RC) -asteapta apasarea unei taste sau combinatii de taste si intoarce un caracter corespunzator;

READLIST (RL) -intoarce o lista care a fost data la intrare; intreaga linie tastata este luata drept lista;

SHOW obiect -tipareste un numar, cuvint sau lista date ca intrare;

SOUND [durata amplitudine] -durata intre 0-255 amplitudinea intre -62 si 75; 0= DO central;

STARTROBOT -pentru un robot atasat calculatorului;

STOPROBOT -operatia inversa lui STARTROBOT;

TYPE obiect1....obiectn -tipareste obiectele pe ecran dar nu lasa o linie libera asa cum face comanda PRINT.

WAIT n -LOGO asteapta n * 1/50 dintr-o secunda;

X. COMENZI ECRAN

+++++++++++++++

BRIGHT N - unde N este 0 sau 1;

CLEARTEXT (CT) - sterge textul de pe ecran; in modul grafic sterge textul din cele doua linii din partea de jos a ecranului;

COPYSCREEN - copiaza ecranul pe ZX printer;

CURSOR - intoarce numarul liniei si coloanei pozitiei cursorului intr-o lista;

FLASH si INVERSE - la fel ca si BRIGHT pina se executa comanda NORMAL;

NORMAL - initializeaza valorile lui FLASH, BRIGHT si INVERSE;

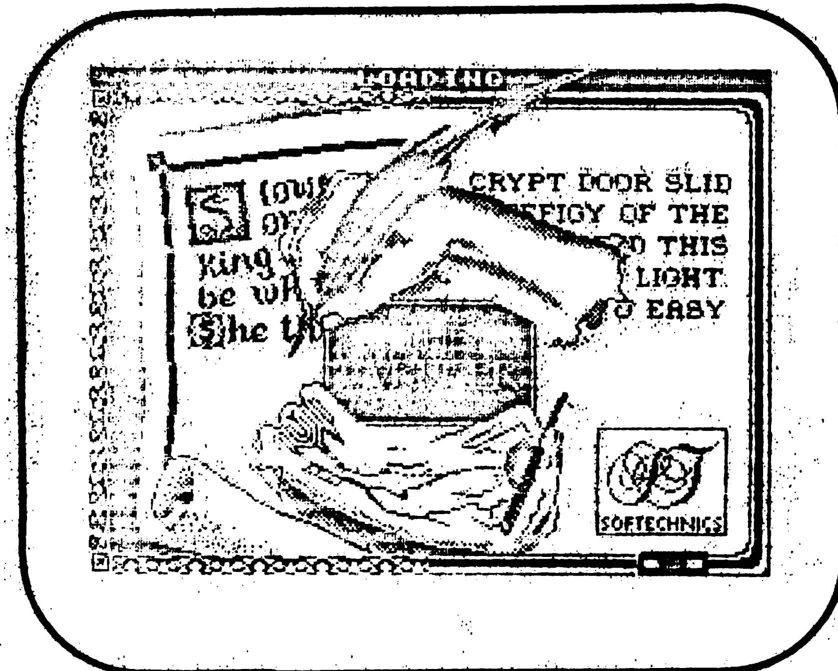
OVER n - la fel ca BRIGHT;

SETCURSOR (SETCUR) [a b] - muta prompterul la coordonatele aratate de a si b ; a intre 0 si 30, b intre 0 si 21;

SETTC [n n] - seteaza culoarea hirtiei si a prim-planului de text;

TEXTSCREEN (TS) - se introduce ecranul de text si se ascunde astfel TURTLE-ul;

TEXTCOLOUR (TC) -seteaza culoare cerneala;



PROCEDURI PRIMITIVE:

OUTPUT (OP) obiect -aceasta primitiva, ca si STOP opreste executia procedurii dar spre deosebire de STOP trimite obiectul inapoi procedurii spre a fi folosit in continuare.

O procedura care, se termina cu END in timpul executiei functioneaza ca o comanda spre deosebire de una care se termina cu OUTPUT si care functioneaza ca o operatie.

REPEAT n listainstr. -repeta o lista de instructiuni de n ori; n trebuie sa fie pozitiv, daca este zecimal se trunchiaza

RUN listainstr. -RUN executa lista de instructiuni ca pe o linie LOGO;

STOP -opreste executia procedurii curente si da controlul procedurii apelante;

TOPLEVEL -opreste executia curenta si preda controlul primei rutine apelante din inlantuire;

XI. SPATIUL DE LUCRU

++++++++++++++++++++

ERALL -sterge tot din spatiul de lucru;

ERASE (ER) nume -sterge procedura specificata de nume;

ERN nume -sterge variabilele specificate, din spatiul de lucru;

ERNS -sterge valorile tuturor variabilelor din spatiul de lucru;

ERPS -sterge toate procedurile;

PO nume -aduce pe ecran definitia procedurii apelate;

POALL -aduce pe ecran titlurile si definitiile procedurilor si valorile pentru fiecare variabila;

PONS -tipareste numele si valorile tuturor variabilelor;

POPS -tipareste definitiile pentru toate procedurile;

POTS -tipareste titlurile procedurilor din spatiul de lucru.

XII. SALVAREA SI INCARCAREA FISIERELOR

++++++++++++++++++++++++++++++++++++++

Un fisier poate fi de trei feluri:

1. fisier procedural -contine proceduri LOGO si eventualele variabile create;
2. fisier editor -constituit din continutul curent al editorului;
3. fisier ecran -ecranul curent.

PROCEDURI PRIMITIVE:

SAVE "numefisier si eventual un nume de obiect;
exemplu: SAVE "fila1 "patrat
SAVE "fila2 [patrat cerc]

SAVEALL "numefisier -salveaza tot ce contine spatiul de lucru in acel moment;

SAVED "numefisier -salveaza continutul editorului

SAVESCR "numefisier -salveaza totul de pe ecran;

SETDRIVE 0...8 -salveaza pe caseta daca este 0 sau pe unul din cele 8 microdrivere;

CATALOG -catalog de microdrive;

ERASEFILE "numefisier tipfisier -pentru MICRO DRIVE -daca tipfisier lipseste se ia implicit tipul LOG. Alte tipuri sint: BIN (binar), SCR (ecran), TXT (editor).

LOAD "numefisier tipfisier -daca doua proceduri au acelasi nume se pastreaza numai cea mai noua;

LOADD "numefisier -incarca tot ce a fost salvat;

LOADSCR "numefisier -incarca si afiseaza ecran;

PRINTON -tipareste pe imprimanta tot ce urmeaza;

PRINTOFF -opreste tiparirea la imprimanta;

COPYSCREEN -copiaza peste 22 linii.

XIII. DEFINIREA SI REDEFINIREA FUNCTIILOR

++

COPYDEF numenou numevechi -copiaza definitia procedurii sub un nou nume;

DEFINE nume lista -are doua intrari: prima un nume de procedura, a doua un nume de lista. Elementele listei constituie intrari pentru procedura specificata, apar astfel ca parametrii ai unei proceduri numele altor proceduri;

DEFINEDP cuvint -intoarce valoarea de adevar TRUE daca cuvintul este numele unei proceduri, altfel FALSE;

PRIMITIVEP cuvint -intoarce TRUE daca cuvintul este numele unei primitive LOGO, altfel FALSE;

TEXT nume -intoarce textul de definire al procedurii specificate de nume, ca pe o lista.

---

exemplu:

```
TO PROCES :X :Y
FD :X
LT :Y
END

PRINT TEXT "PROCES
[ :X :Y ] [ FD :X ] [ LT :Y ]
```

XIV. DIVERSE FUNCTII:

+++++++++++++++++++++

Cu primitivele ce vor fi prezentate se va putea lucra cu memoria direct, deci este bine ca spatiul de lucru sa fie salvat in prealabil, deoarece ar putea fi distrus; in general aceste proceduri incep cu simbolul '.' ;

NODES -intoarce numarul liber de noduri libere, deci spatiul ramas liber pentru variabile, proceduri si rulari. Un nod ocupa 5 octeti in memorie;

RECYCLE -elibereaza nodurile care pot fi eliberate in memorie printr-un proces de curatire a locurilor inutil ocupate din executii anterioare, timpul este de 1 secunda;

.CONTENTS -intoarce o lista cu tot ce intelege compilatorul de LOGO ; aceasta poate ocupa multe noduri in mod inutil;

.PRIMITIVES -tipareste primitivele lui LOGO;

.RESERVE n -daca se doreste rezervarea unui spatiu in memorie pentru niste programe in cod masina trebui la deschiderea sesiunii de lucru sa se introduca numarul de octeti ocupati prin n;

TABELUL STRUCTURII INTERNE A MEMORIEI:

&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

Zona	Adresa
	65535
VARIABILE DE SISTEM LOGO	
	65024
optional: REZERVAT	
	64824
SPATIUL DE LUCRU	
LOGO	
	24832
VARIABILE SISTEM BASIC	
	16384
ROM	
	0

.RESERVED -intoarce adresa de inceput si sfirsit a zonei rezervate in memorie;

.BLOAD "numefisier adresa -incarca fisier in memorie la o adresa dorita; in tabel spre exemplu adresa este 64824.

.BSAVE "numefisier adresa-inceput lungime -salveaza de la adresa respectiva cu lungimea data;

.SETSERIAL n -seteaza viteza de transmisie a datelor ;

.SERIALIN -citeste tot ce soseste la portul serial RS232;

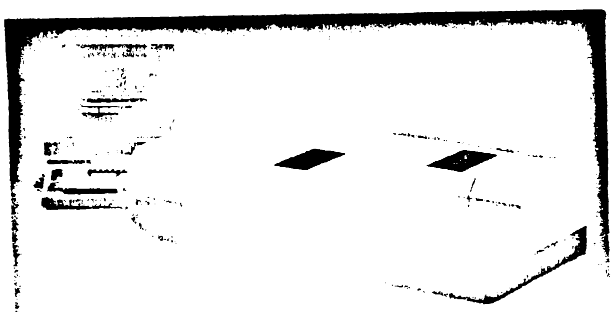
.SERIALOUT n -trimite un octet la portul serial;

.DEPOSIT adresa n -plaseaza valoarea lui n la adresa specificata;

.EXAMINE adresa -intoarce data de la adresa specificata;
.CALL adresa -porneste program in cod masina de la adresa specificata.

XV. MESAJE LOGO:
++++++++++++++++

Not enough inputs to...
I don't know how to...
You don't say what to do with...
...does not output to...
...is used by Logo
...is already defined
...is not true or false
...is not a word
...defined
Too many inside parantheses
...open file problem
...file not found
Bad file name
You're at toplevel
STOPPED!!!
Turtle out of field
Not enough space to proceed
...doesn't like
..has no value
...is a primitive
Not enough items in...
Overflow
...can't devide by zero
...number too big
...as input
...in



DICTIONAR DE PRIMITIVE LOGO:

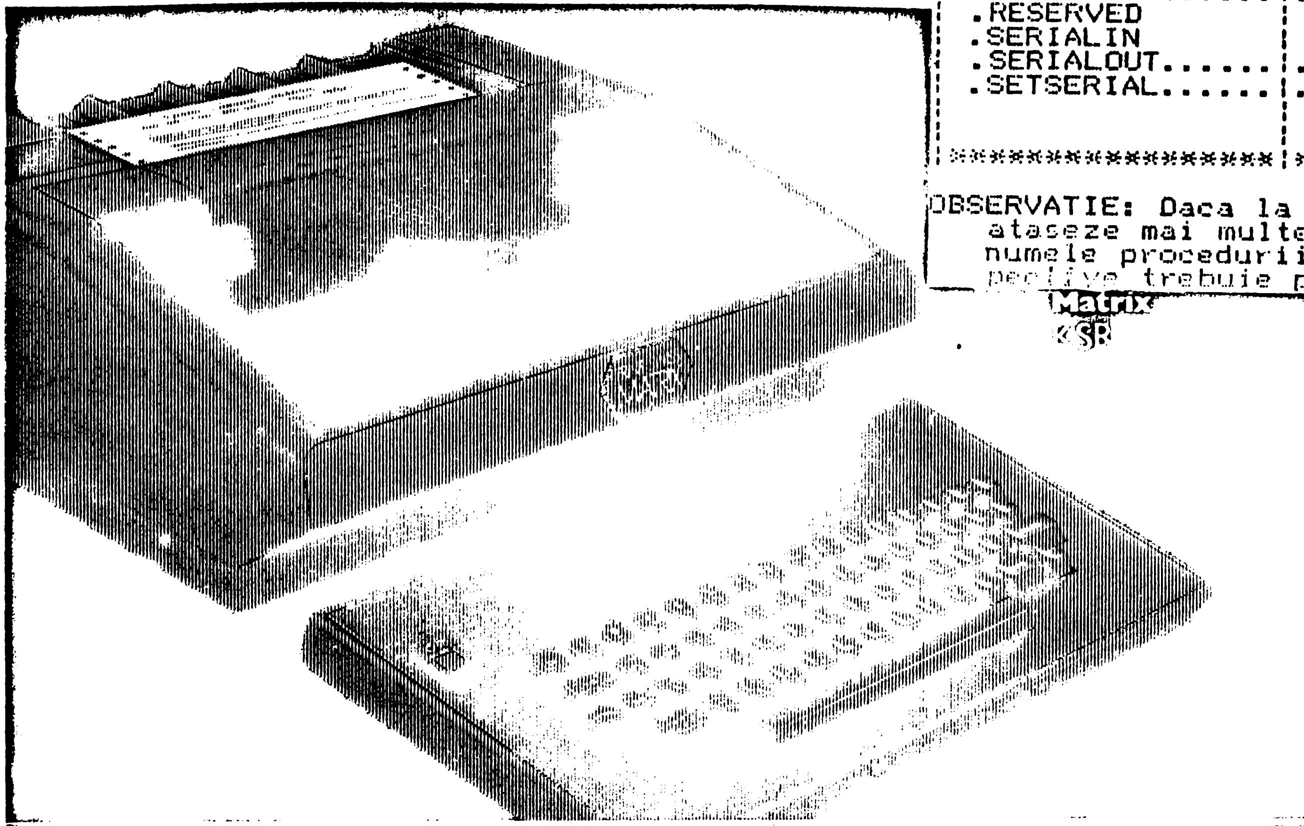
PRIMITIVE	FORMA SCURTA	NECESITA INTRARI
1.Turtle		
BACK	BK	@
BACKGROUND	BG	
CLEAN		
CLEARSCREEN	CS	
DOT		@
FENCE		
FORWARD	FD	@
HEADING		
HIDETURTLE	HT	
HOME		
LEFT	LT	@
PENCOLOUR	PC	
PENDOWN	PD	
PENERASE	PE	
PENREVERSE	PX	
PENUP	PU	
POSITION	POS	
RIGHT	RT	@
SCRUNCH		
SETBG		@
SETBORDER	SETBR	@
SETHEADING	SETH	@
SETPC		@
SETPOS		@
SETSCRUNCH	SETSCR	@
SETX		@
SETY		@
SHOWNP		
SHOWTURTLE	ST	
TOWARDS		
WINDOW		
WRAP		
XCOR		
YCOR		

2.Cuvinte si liste		
ASCII		@
BUTFIRST	BF	@
BUTLAST	BL	@
CHAR		@
COUNT		@
EMPTYP		@
EQUALP		@
FIRST		@
FPUT		@
ITEM		@
LAST		@
LIST		@
LISTP		@
LPUT		@
MEMBERP		@
NUMBERP		@
SENTENCE	SE	@
WORD		@
WORDP		@
3.Variabile		
MAKE		@
NAMEP		@
THING		@
4.Operatii aritmetice		
ARCCOS		@
ARCCOT		@
ARCSIN		@
ARCTAN		@
COSINE	COS	@
COTANGENT	COT	@
DIV		@
INT		@
PRODUCT		@
RANDOM		@
REMAINDER		@
ROUND		@
SINE	SIN	@
SQRT		@
SUM		@
TANGENT	TAN	@
+ - * / = > <		@
5.Definire si editare		
EDIT	ED	@
EDNS		@
TO		@
END		@
6.Expresii conditionale si preluarea controlului		
BYE		
IF		@
OUTPUT	OP	@
REPEAT		@
RUN		@
STOP		
TOPLEVEL		
7.Operatori logici		
AND		@

NOT		@
OR		@
8.Cuvinte externe		
KEYP		
PRINT	PR	@
READCHAR	RC	
READLIST	RL	
SHOW		@
SOUND		@
STARTROBOT		
STOPROBOT		
TYPE		@
WAIT		@
9.Ecran		
BRIGHT		@
CLEARTEXT	CT	@
COPYSCREEN		
CURSOR		
FLASH		@
INVERSE		@
NORMAL		
OVER		@
SETCURSOR	SETCUR	@
SETTC		@
TEXTCOLOUR	TC	
TEXTSCREEN	TS	
10.Spatiul de lucru		
ERALL		
ERASE	ER	@
ERN		@
ERNS		
ERPS		
PO		@
POALL		
PONS		
POPS		
POTS		
11.Salvare si incarcare		
CATALOG		
ERASEFILE		@
LOAD		@
LOADD		@
LOADSCR		@
PRINTON		
PRINTOFF		
SAVE		@
SAVEALL		@
SAVED		@
SAVESCR		@
SETDRIVE		@
12.Functii		
COPYDEF		@
DEFINE		@
DEFINEP		@
PRIMITIVEP		@
TEXT		@

13.Primitive avansate		
NODES		
RECYCLE		
.BLOAD.........		......@
.BSAVE.........		......@
.CALL..........		......@
.CONTENTS		
.DEPOSIT.......		......@
.EXAMINE.......		......@
.PRIMITIVES		
.RESERVE.......		......@
.RESERVED		
.SERIALIN		
.SERIALOUT.....		......@
.SETSERIAL.....		......@
******************	***************	**************

OBSERVATIE: Daca la o procedura trebuie sa se ataseze mai multe intrari (peste doua intrari) numele procedurii impreuna cu intrarile respective trebuie puse in paranteze rotunde.



S.L. ING VOICU MESAROS-ANGHEL
ING. MIODRAG PUTERITY

O MODALITATE PERFORMANTA PENTRU REALIZAREA ANIMATIEI PE CALCULATOARELE COMPATIBILE SPECTRUM

- un pachet de subrutine utile -

0. GENERALITATI

Realizarea animatiei cu ajutorul calculatoarelor s-a dovedit a fi o tehnica atit spectaculoasa cit si dificila. Avind o arie de aplicatie foarte larga, de la simulari in stiinta si tehnica pina la domeniul loisirului (jocuri video, filme) ea a constituit intotdeauna una dintre aplicatiile care folosesc la maximum viteza de calcul si (uneori) memoria disponibila. Prezentul articol isi propune sa dea utilizatorilor calculatoarelor compatibile Spectrum posibilitatea de a realiza animatie la un nivel profesional.

1. PROBLEME

Animatia reprezinta generarea unei imagini in miscare prin afisarea succesiva a unor imagini statice cu o frecventa mai mare decit cea care poate fi decelata de sistemul nervos uman, dind astfel iluzia unei miscari continue. Similar, generarea unei imagini pe un display cu tub catodic se obtine prin baleierea suprafetei ecranului acoperita cu un luminofor de catre un fascicol electronic.

Ne-am pemis sa reamintim aceste considerente binecunoscute pentru ca din ele rezulta trei probleme puse in calea unei animatii de calitate.

1.1. Viteza.

Daca frecventa de afisare a imaginilor statice care compun imaginea in miscare este prea mica se pierde senzatia de miscare continua. Intr-un limbaj de nivel inalt, spre exemplu interpretorul BASIC al Spectrum-ului, viteza de afisare succesiva a imaginilor statice este limitata de viteza de interpretare a unei linii BASIC. Exemplul de mai jos va va edifica:

```
10 FOR X=0 TO 30
20  PRINT AT 11,X;" * "
30 NEXT X
40 FOR X=30 TO 0 STEP -1
```

```
50  PRINT AT 11,X;" * "
60 NEXT X
70 GOTO 10
```

Spatiile din stinga si din dreapta asteriscului sint necesare pentru ca la o noua tiparire vechea pozitie sa fie stearsa. Daca presupunem ca am creste viteza de tiparire (sa zicem prin compilarea acestui program) ceea ce ar deveni suparator ar fi faptul ca distanta intre doua pozitii succesive este prea mare. Apare o a doua problema:

1.2. Distanta intre doua pozitii succesive.

Din Sinclair BASIC putem afisa un caracter doar intr-un cimp de 8*8 pixelei. Acest fapt strica aspectul de miscare "reala" dind senzatia de "artificial" dupa cum se vede si din programul de mai sus. Solutia ar fi sa dispunem de un limbaj care sa permita afisarea unui caracter la oricare din coordonatele celor 256*192 pixeli disponibili pe ecranul Spectrum-ului. Bineinteles pentru ca viteza aparenta de miscare sa ramina aceeasi este necesar ca viteza de afisare sa fie de 8 ori mai mare. Primul deziderat poate fi realizat din Beta BASIC (extensie a Sinclair BASIC-ului binecunoscuta posesorilor de Spectrum):

```
10 FOR X=0 TO 231
20  PLOT X,87;" * "
30 NEXT X
40 FOR X=231 TO 0 STEP -1
50  PLOT X,87;" * "
60 NEXT X
70 GOTO 10
```

In acest caz sintaxa instructiuni PLOT pemite si tiparirea unui sir de caractere in coordonate grafice (in pixeli). Pentru detalii, consultati manualul de Beta BASIC [5]. Prin rularea acestui program se pune in evidenta una dintre cele mai spinoase probleme al unei animatii de calitate si anume ceea ce autorii numesc:

1.3. Fenomenul de pilpiire.

Dupa cum am aratat mai sus, pentru a afisa o noua imagine statica, cea precedenta trebuie stearsa. Atit afisarea imaginii noi cit si stergerea celei vechi consuma un timp finit proportional cu marimea imaginii. In cod masina viteza unui program de tipul celui de mai sus poate fi marita de peste 100 de ori, ceea ce duce la frecvente de afisare comparabile cu cele de baleiere ale fascicolului electronic pe display. In acest caz poate aparea nefericita situatie (dupa Murphy apare sigur) in care fascicolul electronic traverseaza imaginea in miscare tocmai in momentul in care imaginea veche e stearsa si cea noua e in curs de afisare. Daca frecventa de baleiere este egala cu cea de afisare vom vedea doar o imagine incompleta (vertical sau orizontal in functie de modul de afisare), in caz contrar vom avea fenomenul de pilpiire.

O solutie partiala este ca inlocuirea vechii imagini cu cea noua sa nu se faca conform principiului:

- sterge complet imaginea veche
- afiseaza imaginea noua

ci:

- sterge o portiune cit mai mica din imaginea veche
- inlocuieste-o cu cea noua

Aceasta modalitate reduce simtitor efectul de pilpiire (dar nu-l elimina complet) si in plus poate aparea senzatia de "rupere" a imaginii. Aceasta este determinata de faptul ca deasupra zonei in care fascicolul electronic traverseaza imaginea avem imaginea noua iar dedesupt imaginea veche. Pentru a minimaliza acest efect, se poate afisa imaginea statica in sens opus directiei de baleiere (adica se afiseaza de jos in sus si de la dreapta spre stinga). Aceasta metoda a fost utilizata de firma A.C.G. in primele jocuri video din gama ULTIMATE PLAY THE GAME.

2. SOLUTII

Efectul de pilpiire poate fi eliminat complet daca ne asiguram ca fascicolul electronic nu traverseaza niciodata o zona in curs de modificare. Unele calculatoare (ex. COMODORE AMIGA) au facilitati hard care ofera programatorului informatii privind pozitia fascicolului electronic. In acest caz efectul de pilpiire s-ar elimina printr-un rationament de genul:

- asteapta pina cind fascicolul electronic a depasit zona de afisare
- sterge vechea imagine
- afiseaza noua imagine

Un astfel de rationament poate fi cablat hard si duce la asa numita notiune de SPRITE HARD (SPRITE = ceea ce pina acum am

numit imagine statica).

Spectrum-ul nu are astfel de facilitati si sprite-urile trebuie tratate prin soft. Singurul instrument pe care-l avem la indemina este faptul ca putem sti destul de exact cind fascicolul electronic incepe o noua cursa pentru formarea imaginii. Mai cunoastem (ca standarde ale normelor de televiziune) viteza de deplasare a fascicolului electronic, modul de parcurgere a imaginii etc.

Afisarea la nivel de pixeli se poate face scriind un grup de subrutine adecvate acestui scop. Se pune problema gasirii unor solutii ce duc la viteze maxime.

O viteza corespunzatoare se poate obtine numai in cod masina folosind toate artificiile care conduc la cresteri de viteza.

3. METODE PROPUSE PENTRU REALIZAREA ANIMATIEI

In urma numeroaselor experimente care au dus la concluziile de mai sus, autorii au pus la punct un asa numit INTERUPT DRIVEN SPRITE PROCESSOR (pachetul IDSP). Acesta permite utilizarea a opt sprite-uri soft (desi numarul lor poate fi crescut sau scazut in functie de necesitati). Spre deosebire de sprite-urile hard in cazul de fata nu se impun limite privind dimensiunile acestora. Sprite-urile mari insa pot conduce la aparitia unui efect de rupere iar cele foarte mari la un efect de pilpiire.

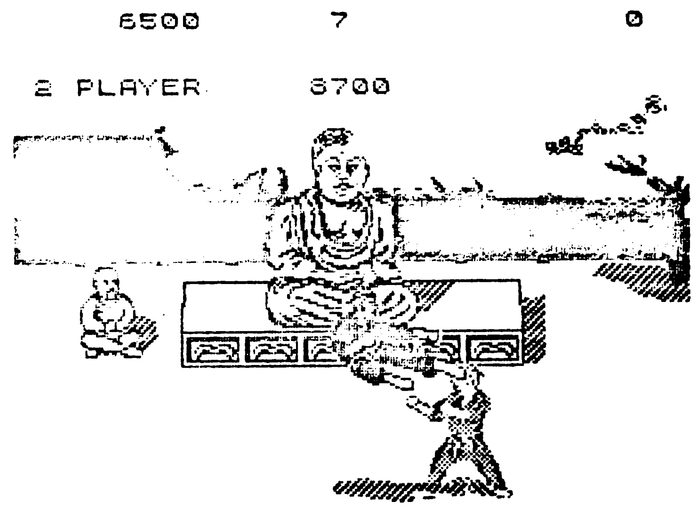
Pachetul IDSP se bazeaza pe faptul ca cererea de intrerupere apare sincron cu impulsul de cadre. S-a utilizat tehnica de vectorizare a intreruperii in IM2 care a fost prezentata pe larg in alt articol al autorilor [2].

Sprite-urile sint afisate in perioada in care fascicolul electonic se afla pe partea superioara a BORDER-ului. Timpul in care are loc afisarea acestora este marcat prin schimbarea culorii border-ului. In mod deliberat, acesta nu a fost limitat la aceasta zona permitind optimizari puternice in cazul unor sprite-uri care-si restring miscarea in zone din partea de jos a ecranului.

Tot din motive de optimizare stergerea sprite-urilor a fost lasata in seama programatorului care trebuie sa prevada un "chenar alb" de dimensiuni mai mari decit distanta maxima intre doua pozitii de afisare succesiva.

Afisarea sprite-urilor se face in modul "intii pe orizontala" in sistemul "natural" (notiuni explicate pe larg in [1]). Astfel se permite o foarte avantajoasa tratare a sprite-urilor care nu au dimensiunea pe verticala ca multiplu de opt pixeli.

Pe orizontala, s-au prevazut opt zone de date pentru acelasi sprite. In cazul unei miscari de alunecare, cele opt zone se pot



construi automat cu ajutorul subrutinei EXPAND. Aparentul consum marit de memorie devine un avantaj cind forma sprite-ului se schimba la deplasarea pe orizontala (ex. mersul omenesc) si cind deplasarea se face cu distante mai mari decit din pixel in pixel.

Viteza maxima la deplasarea din pixel in pixel este de 50 pixeli/sec.

In aceasta versiune s-a eliminat deliberat orice afectare a atributelor de culoare.

Pachetul IDSP este prezentat impreuna cu un program de demonstratie si doua sprite-uri predefinite. De asemenea s-a prevazut si subrutina K_SCAN de baleiere a tastaturii, oferind date pentru programul de demonstratie. Includerea acestei subrutine "in intrerupere" conduce la o "sensibilitate" marita la comanda miscarii sprite-urilor.

4. DESCRIEREA PROGRAMULUI

Programul de demonstratie pernite miscarea a doua sprite-uri. Primul poate fi deplasat atit pe orizontala cit si pe verticala din tastele 6,7,8,9,0 (sau Sinclair joystick). Al doilea se deplaseaza doar pe orizontala la aceeasi coordonata cu

primul. De asemenea din programul de demonstratie se poate observa modul de apelare a subrutinei EXPAND.

Bazele de date pentru cele doua sprite-uri permit evidentierea structurii diferite (miscare de alunecare respectiv miscare complexa).

Subrutinele SPRON si SPROFF permit activarea respectiv dezactivarea procesorului IDSP. Subrutina SPRON mai realizeaza si generarea tabelei de vectorizare a intreruperii [2].

SPROC este apelat la fiecare cerere de intrerupere si afiseaza sprite-urile conform parametrilor din bufferele aferente BUF1 .. BUF8. Afisarea este realizata de subrutina PW care "stie" sa opreasca tiparirea unui sprite de dimensiune nula. Subrutina de calcul a adreselor din display file a fost inclusa in PW din motive de viteza.

K_SCAN si EXPAND sint doua subrutine utile folosite pentru a da respectiv prelua date din/in programul de demonstratie.

In bufferele de sprite-uri BUFn este rezervat, Xn si Yn sint coordonatele de pozitionare (in pixeli), DXn si DYn sint dimensiunile sprite-ului (in sistemul "natural") iar DBADn cele opt adrese corespunzatoare celor opt pozitii pe orizontala. "n" reprezinta numarul de ordine a sprite-ului si este o cifra intre 1 .. 8.

5. CONCLUZII

Metodele propuse au rezultat in urma unui numar mare de variante experimentate. In [3] autorul prezinta un grupaj de metode asemanatoare, care insa se preteaza doar la sprite-uri destinate jocurilor video. In plus, in lucrarea mentionata sint prezente si un numar de erori de programare/conceptie. Autorii prezentului articol au incercat sa largeasca domeniul de utilizare a animatiei inspre latura tehnico-stiintifica si educativa. Dat fiind domeniul relativ larg de utilizare si numarul mic de restrictii impuse, performantele nu sint cele mai ridicate. Autorii au convingerea ca in situatii particulare performantele animatiei pe calculatoarele compatibile Spectrum pot fi imbunatatite sub aspectul marimii imaginilor in miscare.

BIBLIOGRAFIE

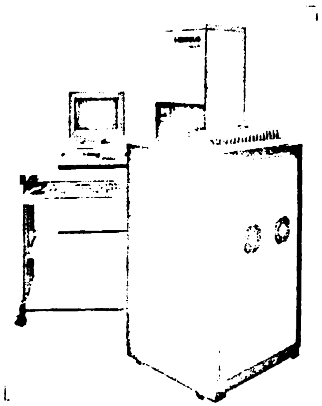
[1] Puterity M., Program pentru vizualizat sprite-uri, seturi de caractere si UDG-uri, Revista INF nr. 1, Timisoara, 1988

[2] Mesaros-Anghel V., Puterity M., Extinderea interpretorului BASIC la computerele compatibile SPECTRUM, Revista INF nr. 2, Timisoara, 1988

[3] Webb, Advanced Spectrum machine language, Melbourne House Publishers, 1983

[4] *** Z80 CPU - instruction set, Zilog

[5] *** Manual de utilizare Beta BASIC, Betasoft



```
HiSoft GEN Assembler ZX Spectrum
ZX Microdrive Version 4.0

Copyright (C) HiSoft 1987
V4.0  All rights reserved

Pass 1 errors: 00

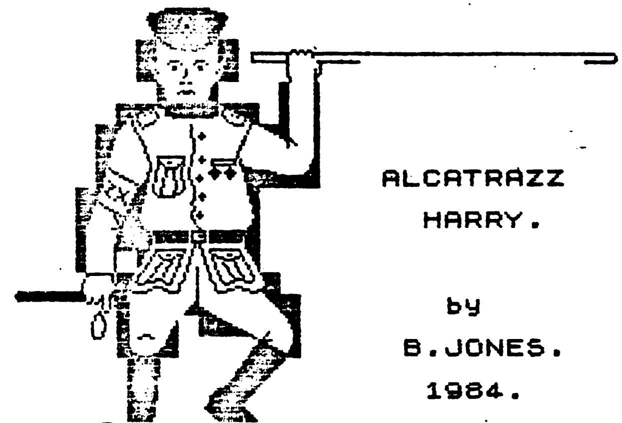
                  100 ************************
                  110 *  ! ! ! D E M O ! !  ! *
                  120 ************************
EA60              130          ORG  60000
EA60              140          ENT  $
EA60 CD73EA       150 DEMO     CALL EXP_N1
EA63 CDBDEA       160          CALL SPRON
EA66 2A24EC       170 DEM1     LD   HL,(X1T)
EA69 22F3F1       180          LD   (X1),HL
EA6C 7D           190          LD   A,L
EA6D 3209F2       200          LD   (X2),A
EA70 18F4         210          JR   DEM1
EA72 C9           220          RET
                  230 ************************
                  240 ;genereaza cele 8 poz.
                  250 ;pe orizontala pt. primul
                  260 ;sprite
EA73 2126EC       270 EXP_N1 LD   HL,DX_N1
EA76 11B9EA       280        LD   DE,DX
EA79 EDA0         290        LDI
EA7B EDA0         300        LDI
EA7D 2128EC       310        LD   HL,NV1_0
EA80 22BBEA       320        LD   (BASE),HL
EA83 CD87EA       330        CALL EXPAND
EA86 C9           340        RET
                  350 ************************
                  360 *          EXPAND         *
                  370 ************************
EA87 2AB9EA       380 EXPAND LD   HL,(DX)
EA8A 45           390        LD   B,L
EA8B 4C           400        LD   C,H
                  410 ;calculate sprite area
EA8C 210000       420        LD   HL,0
```

```
EA8F 59                 LD   E,C
EA90 55                 LD   D,L
EA91 19          AR1    ADD  HL,DE
EA92 10FD               DJNZ AR1
                 ;area in hl
EA94 ED5BBBEA           LD   DE,(BASE)
EA98 19                 ADD  HL,DE
                 ;hl points to base+area
                 ;de points to base
EA99 0607               LD   B,7 ;hor.pos.
EA9B 3ABAEA      EXP3   LD   A,(DY)
EA9E 4F                 LD   C,A
EA9F C5                 PUSH BC
EAA0 3AB9EA      EXP2   LD   A,(DX)
EAA3 47                 LD   B,A
EAA4 05                 DEC  B
                 ;first shift
EAA5 1A                 LD   A,(DE)
EAA6 CB3F               SRL  A
EAA8 77                 LD   (HL),A
EAA9 13                 INC  DE
EAAA 23                 INC  HL
                 ;then rotate
EAAB 1A          EXP1   LD   A,(DE)
EAAC 1F                 RRA
EAAD 77                 LD   (HL),A
EAAE 13                 INC  DE
EAAF 23                 INC  HL
EAB0 10F9               DJNZ EXP1
EAB2 0D                 DEC  C
EAB3 20EB               JR   NZ,EXP2
EAB5 C1                 POP  BC
EAB6 10E3               DJNZ EXP3
EAB8 C9                 RET
                 ************************
EAB9 00          DX     DEFB 0
EABA 00          DY     DEFB 0
EABB 0000        BASE   DEFW 0
                 ************************
EABD 2100FE      SPRON  LD   HL,#FE00
EAC0 01FD00             LD   BC,#00F0
EAC3 71          LP1    LD   (HL),C
EAC4 23                 INC  HL
EAC5 10FC               DJNZ LP1
EAC7 71                 LD   (HL),C
EAC8 3EFE               LD   A,#FE
EACA ED47               LD   I,A
```

```
EACC ED5E               IM   2
EACE FB                 EI
EACF C9                 RET
                 ************************
EAD0 3E3E        SPROFF LD   A,#3E
EAD2 ED56               IM   1
EAD4 ED47               LD   I,A
EAD6 FB                 EI
EAD7 C9                 RET
                 ************************
                 *   SPRITE PROCESSOR    *
                 ************************
EAD8 E5          SPROC  PUSH HL
EAD9 D5                 PUSH DE
EADA C5                 PUSH BC
EADB F5                 PUSH AF
EADC DDE5               PUSH IX
EADE D9                 EXX
EADF C5                 PUSH BC
EAE0 D5                 PUSH DE
EAE1 E5                 PUSH HL
EAE2 D9                 EXX
                 ;border magenta
EAE3 3E03               LD   A,3
EAE5 D3FE               OUT  (254),A
EAE7 DD21F2F1           LD   IX,BUF1
EAEB CD3BEB             CALL PW
EAEE DD2107F2           LD   IX,BUF2
EAF2 CD3BEB             CALL PW
EAF5 DD211CF2           LD   IX,BUF3
EAF9 CD3BEB             CALL PW
EAFC DD2121F2           LD   IX,BUF4
EB00 CD3BEB             CALL PW
EB03 DD2126F2           LD   IX,BUF5
EB07 CD3BEB             CALL PW
EB0A DD212BF2           LD   IX,BUF6
EB0E CD3BEB             CALL PW
EB11 DD2130F2           LD   IX,BUF7
EB15 CD3BEB             CALL PW
EB18 DD2135F2           LD   IX,BUF8
EB1C CD3BEB             CALL PW
                 ************************
                 ;border red
EB1F 3E02               LD   A,2
EB21 D3FE               OUT  (254),A
EB23 CD3BEB             CALL K_SCAN
                 ;border magenta
EB26 3E03               LD   A,3
```

```
EB28 D3FE               OUT  (254),A
                 ************************
EB2A D9                 EXX
EB2B E1                 POP  HL
EB2C D1                 POP  DE
EB2D C1                 POP  BC
EB2E D9                 EXX
EB2F DDE1               POP  IX
EB31 F1                 POP  AF
EB32 C1                 POP  BC
EB33 D1                 POP  DE
EB34 E1                 POP  HL
EB35 FB                 EI
                 ;border green
EB36 3E04               LD   A,4
EB38 D3FE               OUT  (254),A
EB3A C9                 RET
                 **********************
                 *     PRINT WINDOW     *
                 **********************
EB3B DD4604      PW     LD   B,(IX+4);dy
EB3E 78                 LD   A,B
EB3F B7                 OR   A
EB40 C8                 RET  Z
                 ;self modifying code
EB41 DD5601             LD   D,(IX+1);x
EB44 DD5E02             LD   E,(IX+2);y
EB47 7A                 LD   A,D
EB48 E607               AND  %00000111
EB4A 17                 RLA
EB4B C605               ADD  A,5
EB4D 3257EB             LD   (I1+2),A
EB50 3C                 INC  A
EB51 325AEB             LD   (I2+2),A
EB54 D9                 EXX
                 ; modifys here
EB55 DD5E00      I1     LD   E,(IX+0)
EB58 DD5600      I2     LD   D,(IX+0)
                 ; de' pointer in dbase
EB5B D9                 EXX
EB5C D9          PW3    EXX
EB5D DD4603             LD   B,(IX+3)
EB60 D9                 EXX
EB61 7A                 LD   A,D
EB62 1F                 RRA
EB63 1F                 RRA
EB64 1F                 RRA
EB65 E61F               AND  %00011111
```

```
EB67 6F       1860          LD   L,A
EB68 7B       1870          LD   A,E
EB69 17       1880          RLA
EB6A 17       1890          RLA
EB6B E6E0     1900          AND  %11100000
EB6D B5       1910          OR   L
EB6E 6F       1920          LD   L,A
EB6F 7B       1930          LD   A,E
EB70 1F       1940          RRA
EB71 1F       1950          RRA
EB72 1F       1960          RRA
EB73 E618     1970          AND  %00011000
EB75 67       1980          LD   H,A
EB76 7B       1990          LD   A,E
EB77 E607     2000          AND  %00000111
EB79 F640     2010          OR   %01000000
EB7B B4       2020          OR   H
EB7C 67       2030          LD   H,A
EB7D E5       2040          PUSH HL
EB7E D9       2050          EXX
EB7F E1       2060          POP  HL
EB80 1A       2070 PW2      LD   A,(DE)
EB81 77       2080          LD   (HL),A
EB82 2C       2090          INC  L
EB83 13       2100          INC  DE
EB84 10FA     2110          DJNZ PW2
EB86 D9       2120          EXX
EB87 1C       2130          INC  E
EB88 10D2     2140          DJNZ PW3
EB8A C9       2150          RET
              2160 ************************
              2170 *        KEY SCAN      *
              2180 ************************
EB8B 01FEEF   2190 K_SCAN LD   BC,61438
              2200 ;Sinclair joystick 1
EB8E ED48     2210          IN   C,(C)
EB90 79       2220          LD   A,C
EB91 E61E     2230          AND  %00011110
EB93 EE0C     2240          XOR  %00001100
EB95 2851     2250          JR   Z,UL
EB97 79       2260          LD   A,C
EB98 E61E     2270          AND  %00011110
EB9A EE14     2280          XOR  %00010100
EB9C 2859     2290          JR   Z,UR
EB9E 79       2300          LD   A,C
EB9F E61E     2310          AND  %00011110
EBA1 EE0A     2320          XOR  %00001010
EBA3 2861     2330          JR   Z,DL
EBA5 79       2340          LD   A,C
EBA6 E61E     2350          AND  %00011110
EBA8 EE12     2360          XOR  %00010010
EBAA 2869     2370          JR   Z,DR
EBAC 79       2380          LD   A,C
EBAD CB47     2390          BIT  0,A ;FIRE
EBAF 2811     2400          JR   Z,FIRE
EBB1 CB4F     2410          BIT  1,A
EBB3 2813     2420          JR   Z,UP
EBB5 CB57     2430          BIT  2,A
EBB7 2817     2440          JR   Z,DOWN
EBB9 CB5F     2450          BIT  3,A
EBBB 281B     2460          JR   Z,RIGHT
EBBD CB67     2470          BIT  4,A
EBBF 281F     2480          JR   Z,LEFT
EBC1 C9       2490          RET
              2500 ;
EBC2 CDD0EA   2510 FIRE     CALL SPROFF
EBC5 C37075   2520          JP   30064
              2530 ;will be modified approp.
              2540 ;
EBC8 3A25EC   2550 UP       LD   A,(Y1T)
EBCB 3D       2560          DEC  A
EBCC 3225EC   2570          LD   (Y1T),A
EBCF C9       2580          RET
EBD0 3A25EC   2590 DOWN     LD   A,(Y1T)
EBD3 3C       2600          INC  A
EBD4 3225EC   2610          LD   (Y1T),A
EBD7 C9       2620          RET
EBD8 3A24EC   2630 RIGHT    LD   A,(X1T)
EBDB 3C       2640          INC  A
EBDC 3224EC   2650          LD   (X1T),A
EBDF C9       2660          RET
EBE0 3A24EC   2670 LEFT     LD   A,(X1T)
EBE3 3D       2680          DEC  A
EBE4 3224EC   2690          LD   (X1T),A
EBE7 C9       2700          RET
EBE8 3A24EC   2710 UL       LD   A,(X1T)
EBEB 3D       2720          DEC  A
EBEC 3224EC   2730          LD   (X1T),A
EBEF 3A25EC   2740          LD   A,(Y1T)
EBF2 3D       2750          DEC  A
EBF3 3225EC   2760          LD   (Y1T),A
EBF6 C9       2770          RET
EBF7 3A24EC   2780 UR       LD   A,(X1T)
EBFA 3C       2790          INC  A
EBFB 3224EC   2800          LD   (X1T),A
EBFE 3A25EC   2810          LD   A,(Y1T)
```



ALCATRAZZ HARRY.

by B.JONES. 1984.

© MASTERTRONIC LTD.

```
EC01 3D       2820          DEC  A
EC02 3225EC   2830          LD   (Y1T),A
EC05 C9       2840          RET
EC06 3A24EC   2850 DL       LD   A,(X1T)
EC09 3D       2860          DEC  A
EC0A 3224EC   2870          LD   (X1T),A
EC0D 3A25EC   2880          LD   A,(Y1T)
EC10 3C       2890          INC  A
EC11 3225EC   2900          LD   (Y1T),A
EC14 C9       2910          RET
EC15 3A24EC   2920 DR       LD   A,(X1T)
EC18 3C       2930          INC  A
EC19 3224EC   2940          LD   (X1T),A
EC1C 3A25EC   2950          LD   A,(Y1T)
EC1F 3C       2960          INC  A
EC20 3225EC   2970          LD   (Y1T),A
EC23 C9       2980          RET
              2990 ************************
EC24 00       3000 X1T      DEFB 0
EC25 00       3010 Y1T      DEFB 0
              3020 ************************
              3030 ************************
              3040 *  NAVA 1 (7*18,MOD 2)  *
              3050 ************************
EC26 07       3060 DX_N1    DEFB 7
EC27 14       3070 DY_N1    DEFB 20
```

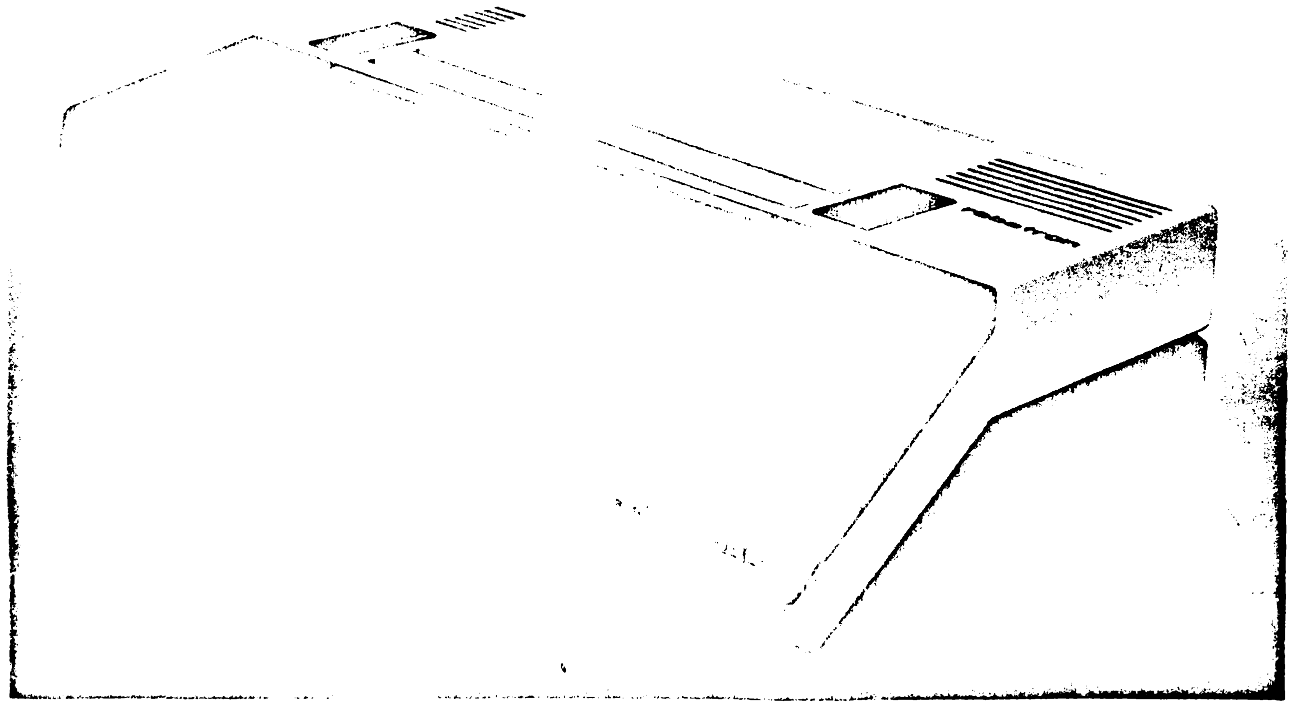
```
EC28 00000000  3080 NVI_0  DEFB 0,0,0,0,0,0,0,31,254,0,0,0,0,0,31,255,0,0,0,0,0
EC3D 19C38000  3090        DEFB 25,195,128,0,0,0,0,31,255,192,0,0,0,0
EC4B 0FFFE000  3100        DEFB 15,255,224,0,0,0,0,14,3,255,0,0,0,0
EC59 0E77C7DF  3110        DEFB 14,119,199,223,240,0,0,7,239,207,255,12,0,0
EC67 07DCDF57  3120        DEFB 7,220,223,87,3,0,0,0,128,254,175,[illegible],[illegible],0
EC75 0CFFFEAF  3130        DEFB 12,255,254,175,255,191,0,28,130,27,[illegible],255,188,0
EC83 7FFFFFFF  3140        DEFB 127,255,255,255,3,224,0,127,252,15,28,0,192,0
EC91 1C700780  3150        DEFB 28,112,7,128,0,0,0,0,0,3,243,128,0,0
EC9F 000000FF  3160        DEFB 0,0,0,255,248,0,0,0,0,1,243,128,0,0,0,0,0,0,0,0,
0
ECB4           3170 NV1_1  DEFS 7*20
ED40           3180 NV1_2  DEFS 7*20
EDCC           3190 NV1_3  DEFS 7*20
EE58           3200 NV1_4  DEFS 7*20
EEE4           3210 NV1_5  DEFS 7*20
EF70           3220 NV1_6  DEFS 7*20
EFFC           3230 NV1_7  DEFS 7*20
               3240 ***********************
               3250 * ROBOT ROTITOR (3*15)M2*
               3260 *  NUMAI PE ORIZONTALA  *
               3270 ***********************
F088 03        3280 DX_RR  DEFB 3
F089 0F        3290 DY_RR  DEFB 15
F08A 03E0000F  3300 RR_0   DEFB 3,224,0,15,248,0,31,252,0,63,254,0
F096 3FFE007F  3310        DEFB 63,254,0,127,255,0,127,255,0,127,255,0
F0A2 7FFF007F  3320        DEFB 127,255,0,127,255,0,63,254,0,63,254,0
F0AE 1FFC000F  3330        DEFB 31,252,0,15,248,0,3,224,0
F0B7 01F00007  3340 RR_1   DEFB 1,240,0,7,124,0,14,254,0,28,255,0
F0C3 1DFF0039  3350        DEFB 29,255,0,57,255,128,63,255,128,59,255,128
F0CF 3FFF8039  3360        DEFB 63,255,128,57,255,128,29,255,0,28,255,0
F0DB 0EFE0007  3370        DEFB 14,254,0,7,124,0,1,240,0
F0E4 00F80003  3380 RR_2   DEFB 0,248,0,3,222,0,7,223,0,15,143,128
F0F0 0FDF801F  3390        DEFB 15,223,128,31,143,192,31,255,192,31,223,192
F0FC 1FFFC01F  3400        DEFB 31,255,192,31,143,192,15,223,128,15,143,128
F108 07DF0003  3410        DEFB 7,223,0,3,222,0,0,248,0
F111 007C0001  3420 RR_3   DEFB 0,124,0,1,247,0,3,251,128,7,249,192
F11D 07FDC00F  3430        DEFB 7,253,192,15,252,224,15,255,224,15,254,224
F129 0FFFE00F  3440        DEFB 15,255,224,15,252,224,7,253,192,7,249,192
F135 03FB8001  3450        DEFB 3,251,128,1,247,0,0,124,0
F13E 003E0000  3460 RR_4   DEFB 0,62,0,0,255,128,1,255,192,3,15,224
F14A 02D7E006  3470        DEFB 2,215,224,6,151,240,7,111,240,7,255,240
F156 07FFF007  3480        DEFB 7,255,240,7,255,240,3,255,224,3,255,224
F162 01FFC000  3490        DEFB 1,255,192,0,255,128,0,62,0
F16B 001F0000  3500 RR_5   DEFB 0,31,0,0,127,192,0,255,224,1,224,240
F177 01DF7003  3510        DEFB 1,223,112,3,219,120,3,228,248,3,255,248
F183 03FFF803  3520        DEFB 3,255,248,3,255,248,1,255,240,1,255,240
F18F 00FFE000  3530        DEFB 0,255,224,0,127,192,0,31,0
F198 000F8000  3540 RR_6   DEFB 0,15,128,0,63,224,0,127,240,0,254,24,0,253,104
F1A7 01FD2C01  3550        DEFB 1,253,44,1,254,220,1,255,252,1,255,252,1,255,252
F1B6 00FFF800  3560        DEFB 0,255,248,0,255,248,0,127,240,0,63,224,0,15,128
F1C5 0007C000  3570 RR_7   DEFB 0,7,192,0,31,240,0,63,248,0,127,252
F1D1 007FFC00  3580        DEFB 0,127,252,0,255,254,0,255,254,0,255,254
F1DD 00FFFE00  3590        DEFB 0,255,254,0,255,254,0,127,252,0,127,252
F1E9 003FF800  3600        DEFB 0,63,248,0,31,240,0,7,192
               3610 ***********************
               3620 ***********************
               3630 * SPRITES DATA BUFFERS *
               3640 ***********************
F1F2 00        3650 BUF1   DEFB 0
F1F3 00        3660 X1     DEFB 0
F1F4 00        3670 Y1     DEFB 0
F1F5 07        3680 DX1    DEFB 7;dx_n1
F1F6 14        3690 DY1    DEFB 20;dy_n1
F1F7 28ECB4EC  3700 DBAD1  DEFW NV1_0,NV1_1,NV1_2,NV1_3,NV1_4,NV1_5,NV1_6,NV1_7
F207 00        3710 BUF2   DEFB 0
F208 00        3720 X2     DEFB 0
F209 18        3730 Y2     DEFB 24;pt. demo
F20A 03        3740 DX2    DEFB 3;dx_rr
F20B 0F        3750 DY2    DEFB 15;dy_rr
F20C 8AF0B7F0  3760 DBAD2  DEFW RR_0,RR_1,RR_2,RR_3,RR_4,RR_5,RR_6,RR_7
               3770 ;
               3780 ;urmatoarele 6 buffere
               3790 ;sint nefolosite in
               3800 ;aceasta demonstratie
               3810 ;
F21C 00        3820 BUF3   DEFB 0
F21D 00        3830 X3     DEFB 0
F21E 00        3840 Y3     DEFB 0
F21F 00        3850 DX3    DEFB 0
F220 00        3860 DY3    DEFB 0
               3870 ;DBAD3  DEFW DB1,DB2,DB3,DB4,DB5,DB6,DB7,DB8
F221 00        3880 BUF4   DEFB 0
F222 00        3890 X4     DEFB 0
F223 00        3900 Y4     DEFB 0
F224 00        3910 DX4    DEFB 0
F225 00        3920 DY4    DEFB 0
               3930 ;DBAD4  DEFW DB1,DB2,DB3,DB4,DB5,DB6,DB7,DB8
F226 00        3940 BUF5   DEFB 0
F227 00        3950 X5     DEFB 0
F228 00        3960 Y5     DEFB 0
F229 00        3970 DX5    DEFB 0
F22A 00        3980 DY5    DEFB 0
               3990 ;DBAD5  DEFW DB1,DB2,DB3,DB4,DB5,DB6,DB7,DB8
F22B 00        4000 BUF6   DEFB 0
F22C 00        4010 X6     DEFB 0
F22D 00        4020 Y6     DEFB 0
F22E 00        4030 DX6    DEFB 0
```

```
F22F 00          4040 DY6      DEFB 0
                 4050 ;DBAD6   DEFW DB1,DB2,DB3,DB4,DB5,DB6,DB7,DB8
F230 00          4060 BUF7     DEFB 0
F231 00          4070 X7       DEFB 0
F232 00          4080 Y7       DEFB 0
F233 00          4090 DX7      DEFB 0
F234 00          4100 DY7      DEFB 0
                 4110 ;DBAD7   DEFW DB1,DB2,DB3,DB4,DB5,DB6,DB7,DB8
F235 00          4120 BUF8     DEFB 0
F236 00          4130 X8       DEFB 0
F237 00          4140 Y8       DEFB 0
F238 00          4150 DX8      DEFB 0
F239 00          4160 DY8      DEFB 0
                 4170 ;DBAD8   DEFW DB1,DB2,DB3,DB4,DB5,DB6,DB7,DB8
                 4180 ************************
FDFD             4190          ORG  #FDFD
FDFD C3D8EA      4200          JP   SPROC
                 4210 ************************
```

```
Pass 2 errors: 00

AR1     EA91    BASE    EABB
BUF1    F1F2    BUF2    F207
BUF3    F21C    BUF4    F221
BUF5    F226    BUF6    F22B
BUF7    F230    BUF8    F235
DBAD1   F1F7    DBAD2   F20C
DEM1    EA66    DEMO    EA60
DL      EC06    DOWN    EBD0
DR      EC15    DX      EAB9
DX1     F1F5    DX2     F20A
DX3     F21F    DX4     F224
DX5     F229    DX6     F22E
DX7     F233    DX8     F238
DX_N1   EC26    DX_RR   F088
DY      EABA    DY1     F1F6
DY2     F20B    DY3     F220
DY4     F225    DY5     F22A
DY6     F22F    DY7     F234
DY8     F239    DY_N1   EC27
DY_RR   F089    EXP1    EAAB
EXP2    EAA0    EXP3    EA9B
EXPAND  EA87    EXP_N1  EA73
FIRE    EBC2    I1      EB55
I2      EB58    K_SCAN  EB3B
LEFT    EBE0    LP1     EAC3
NV1_0   EC28    NV1_1   ECB4
NV1_2   ED40    NV1_3   EDCC
NV1_4   EE58    NV1_5   EEE4
NV1_6   EF70    NV1_7   EFFC
PW      EB3B    PW2     ED80
PW3     EB5C    RIGHT   EBD3
RR_0    F08A    RR_1    F0B7
RR_2    F0E4    RR_3    F111
RR_4    F13E    RR_5    F16B
RR_6    F198    RR_7    F1C5
SPROC   EAD8    SPROFF  EAD0
SPRON   EABD    UL      EBE8
UP      EBC8    UR      EBF7
X1      F1F3    X1T     EC24
X2      F208    X3      F21D
X4      F222    X5      F227
X6      F22C    X7      F231
X8      F236    Y1      F1F4
Y1T     EC25    Y2      F209
Y3      F21E    Y4      F223
Y5      F228    Y6      F22D
Y7      F232    Y8      F237

Table used:  998  from  1031
Executes: 60000
```



RADU DRAGOMIR
LIST-ROMOM

```
*HISOFT GENS3M2 ASSEMBLER*
      ZX SPECTRUM

Copyright (C) HISOFT 1983,4
All rights reserved

Pass 1 errors: 00

        10 *C-
        20 ;PROGRAM DE LISTARE
        30 ;PE IMPRIMANTA ROMOM
        40
        50
        60 ;C 1988 RADU DRAGOMIR
        70
        80
        90
       100 ;AVANTAJE:
       110
       120 ;-ESTE RELOCATABIL
       130 ;-FOLOSESTE INTREAGA
       140 ;LUNGIME A RINDULUI
       150 ;-SE POATE SPECIFICA
       160 ;MARGINEA STINGA CU
       170 ;POKE X+50,N:LPRINT
       180 ;X ADR DE INCARCARE
       190 ;N NR DE SPATII
       200 ;-SE LANSEAZA CU
       210 ;RANDOMIZE USR X
       220 ;********************
       230
C2C3   240          PUSH BC
C2C4   250          LD   A,#E4
C2C6   260          CALL #38D4
C2C9   270          LD   A,#AD
C2CB   280          CALL #38D4
C2CE   290          LD   A,#FF
C2D0   300          CALL #38D4
C2D3   310          POP  BC
C2D4   320          LD   HL,#0019
C2D7   330          ADD  HL,BC
C2D8   340          LD   (#5CC5),H
L
C2DB   350          RET
       360
C2DC   370          CP   #80
C2DE   380          JR   NC,ET3
C2E0   390          CP   #31
C2E2   400          JR   C,ET1
C2E4   410 ET2      CPL
C2E5   420          JP   #38D4
C2E8   430 ET1      CP   #0D
C2EA   440          RET  NZ
C2EB   450          CPL
C2EC   460          CALL #38D4
C2EF   470          LD   A,#F5
C2F1   480          CALL #38D4
C2F4   490          LD   B,6
C2F6   500 REL      PUSH BC
C2F7   510          LD   A,#DF
C2F9   520          CALL #38D4
C2FC   530          POP  BC
C2FD   540          DJNZ REL
C2FF   550          RET
C300   560 ET3      SUB  #A5
C302   570          JP   NC,#0C10
C305   580          LD   A,#20
C307   590          JR   ET2
       600
       610 ;********************

Pass 2 errors: 00

Table used:    53  from   199
```

OVIDIU ANDRASESCU
TEMA DE CASA

SE DA: O SECRETARA LA MAI MULTI DIRECTORI, SECRETARA CARE ARE: UN CALCULATOR UN HARDIST SI UN SOFTIST, PERSONALE.

SE CERE: SA SE AJUTE SECRETARA; SA I SE PUNA LA DISPOZITIE O MINICENTRALA TELEFONICA AVIND URMATOARELE POSIBILITATI:
- AGENDA, CU NUME, ADRESE SI NUMERE DE TELEFON;
- APELEUL (DIRECT IN RETEAUA TELEFONICA, IN IMPULSURI SPECIFICE) A CORESPONDENTULUI, PRIN NUME SAU ADRESA SAU NUMAR DE TELEFON SAU NUMAR DE ORDINE DIN AGENDA;
- APELUL NUMARULUI (SAU NUMERELOR DIN COADA DE ASTEPTARE, PE RIND), PINA CIND RASPUNDE;
- APELUL CORESPONDENTULUI CU TEXTUL (AUDIO) "ALO, VA ROG ASTEPTATI" SI SEMNALIZAREA CONTACTARII;
- LA APELURI DIN EXTERIOR, ACELASI TEXT PENTRU CEL CE SUNA SI ACEIASI SEMNALIZARE;

SE DA: UN TATIC, CALCULATORIST, DOTAT EVIDENT CU UN CALCULATOR.

SE CERE: REALIZAREA UNUI PROGRAM CE PERMITE INVATAREA CITIRII, ASTFEL: LA TASTARE UNEI LITERE, APARE SCRISA MARE PE ECRAN SI SE AUDE TARE IN DIFUZOR.

SE DA: UN TEXT DE DOCUMENTATIE, UN CALCULATOR SI O IMPRIMANTA MATRICIALA.

SE CERE: SA SE REALIZEZE UN EDITOR DE TEXTE CARE SA STIE SA SCRIE UN NUMAR VARIABIL DE CARACTERE PE RIND, ASTFEL INCIT UN RIND DE "I"-URI SA AIBA MULT MAI MULTE CARACTERE DECIT UN RIND DE "W"-URI.

AȘTEPTĂM SUGESTIILE DUMNEAVOASTRĂ
ȘI MATERIALELE PROPUSE SPRE PUBLICARE, LA ADRESA:

CASA UNIVERSITARILOR
str. Paris nr. 1
1900 Timișoara

INF nr.2/1988

Lei 35